2018-03

Network anomaly detection with stable distributions

Bollmann, C.A.

Monterey, California. Naval Postgraduate School

NAVAL POSTGRADUATE SCHOOL

MONTEREY, CALIFORNIA

DISSERTATION

NETWORK ANOMALY DETECTION WITH STABLE DISTRIBUTIONS

by

C. A. Bollmann

March 2018

Dissertation Co-Supervisors:	Murali Tummala
	John McEachen

REPORT DOCUMENTATION PAGE	*Form Approved OMB No. 0704–0188*

1. AGENCY USE ONLY *(Leave Blank)*	2. REPORT DATE March 2018	3. REPORT TYPE AND DATES COVERED Doctoral Dissertation 07-13-2016 to 02-07-2018

4. TITLE AND SUBTITLE NETWORK ANOMALY DETECTION WITH STABLE DISTRIBUTIONS	5. FUNDING NUMBERS
6. AUTHOR(S) C.A. Bollmann	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey, CA 93943	8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES) Laboratory for Telecommunication Sciences 8080 Greenmeade Drive College Park, MD 20740	10. SPONSORING / MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES

The views expressed in this document are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government. IRB Protocol Number: N/A.

12a. DISTRIBUTION / AVAILABILITY STATEMENT Approved for public release. Distribution is unlimited.	12b. DISTRIBUTION CODE

13. ABSTRACT *(maximum 200 words)*

Network anomaly detection must be automated to meet requirements for real-time, accurate monitoring in the face of exponentially-growing traffic volumes; however, this accuracy is often reduced when Gaussian methods are applied to non-Gaussian network traffic. To improve detection accuracy at requisite low false-alarm rates, we propose modeling network traffic and detecting anomalies using an entirely non-Gaussian methodology based on the α-stable distribution and appropriately-derived stable estimators.

Using three publicly-available network traffic traces, we show that the non-Gaussian stable distribution provides a more accurate traffic model under benign and attack scenarios, as well as a mixture of these conditions. In this research, we demonstrate that an α-stable traffic model enables adaptive techniques while significantly reducing data fit errors. To improve the accuracy of anomaly detection, computationally-efficient, α-stable -derived location and dispersion estimators are identified and developed. These estimators are implemented in a novel proof-of-concept, non-parametric, non-Gaussian detection system based on α-stable principles. The proposed real-time detection system achieves higher accuracy at a lower error rate than equivalent Gaussian methods and comparable state-of-the-practice techniques.

14. SUBJECT TERMS alpha stable distribution, statistical network anomaly detection, non-parametric anomaly detection, levy location, sample myriad, zero order statistics, zero order location, zero order dispersion, non-Gaussian detection, heavy-tailed detection, alpha-stable network anomaly detection, stable attribute estimator	15. NUMBER OF PAGES 177
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	UU

NSN 7540-01-280-5500 Standard Form 298 (Rev. 2–89) Prescribed by ANSI Std. 239–18

THIS PAGE INTENTIONALLY LEFT BLANK

NETWORK ANOMALY DETECTION WITH STABLE DISTRIBUTIONS

C. A. Bollmann
Commander, United States Navy
S.M., Nuclear Engineering, Massachusetts Institute of Technology, 1998
S.M., Technology & Policy, Massachusetts Institute of Technology, 1998

Submitted in partial fulfillment of the
requirements for the degree of

DOCTOR OF PHILOSOPHY IN ELECTRICAL AND COMPUTER ENGINEERING

from the

NAVAL POSTGRADUATE SCHOOL
March 2018

Approved by:	Murali Tummala Professor of Electrical and Computer Engineering Dissertation Co-Supervisor Dissertation Committee Chair	John McEachen Professor of Electrical and Computer Engineering Dissertation Co-Supervisor
	Douglas Fouts Professor of Electrical and Computer Engineering	James Scrofani Professor of Electrical and Computer Engineering
	Robert Beverly Professor of Computer Science	
Approved by:	R. Clark Robertson Chair, Department of Electrical and Computer Engineering	
Approved by:	Douglas Moses Vice Provost for Academic Affairs	

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

Network anomaly detection must be automated to meet requirements for real-time, accurate monitoring in the face of exponentially-growing traffic volumes; however, this accuracy is often reduced when Gaussian methods are applied to non-Gaussian network traffic. To improve detection accuracy at requisite low false-alarm rates, we propose modeling network traffic and detecting anomalies using an entirely non-Gaussian methodology based on the α-stable distribution and appropriately-derived stable estimators.

Using three publicly-available network traffic traces, we show that the non-Gaussian stable distribution provides a more accurate traffic model under benign and attack scenarios, as well as a mixture of these conditions. In this research, we demonstrate that an α-stable traffic model enables adaptive techniques while significantly reducing data fit errors. To improve the accuracy of anomaly detection, computationally-efficient, α-stable -derived location and dispersion estimators are identified and developed. These estimators are implemented in a novel proof-of-concept, non-parametric, non-Gaussian detection system based on α-stable principles. The proposed real-time detection system achieves higher accuracy at a lower error rate than equivalent Gaussian methods and comparable state-of-the-practice techniques.

THIS PAGE INTENTIONALLY LEFT BLANK

Contents

List of Figures

THIS PAGE INTENTIONALLY LEFT BLANK

List of Tables

THIS PAGE INTENTIONALLY LEFT BLANK

List of Acronyms and Abbreviations

AD	anomaly detection
ARFIMA	auto-regressive fractional integrated moving average
AUC	area under the curve
CFAR	constant false alarm rate
CUSUM	cumulative sum
DDoS	distributed denial-of-service
DoS	denial-of-service
EWMA	exponentially-weighted moving average
FAR	false alarm rate
FLOS	fractional lower-order statistics
FN	false negative
FP	false positive
FPR	false positive rate
GHz	gigahertz
Gbps	gigabit per second
GLRT	generalized likelihood ratio test
GMT	Greenwich Mean Time
HHH	hierarchical heavy hitters
IP	internet protocol

ISCX	Information Security Center of Excellence
LL	log likelihood
LLE	Lévy location estimate
LLR	log-likelihood ratio
LLRT	log-likelihood ratio test
LMA	Lévy mixture approximation
LRT	likelihood ratio test
MACCDC	Mid-Atlantic Collegiate Cyber-Defense Competition
MAWI	Measurement and Analysis on the Wide Internet
ML	maximum-likelihood
MLE	maximum likelihood estimation
MOC	moment-order constraint
PaS	positive α-stable
PDF	probability density function
PSS	probabilistic sample smoothing
ROC	receiver operating characteristic
RV	random variable
SAE	stable attribute estimator
SaS	symmetric α-stable
SE	stable estimator
SNR	signal-to-noise ratio
TCP	Transmission Control Protocol

TP	true positive
TPR	true positive rate
UDP	user datagram protocol
ZOD	zero-order dispersion
ZOL	zero-order location
ZOP	zero-order power
ZOS	zero-order statistics

THIS PAGE INTENTIONALLY LEFT BLANK

Acknowledgments

Finishing this dissertation leaves me perpetually grateful, for I have required many shoulders in order to draw this research to a close.

Those shoulders include:
My wife and girls for being my biggest cheerleaders, and happily (or at least understandingly) taking care of business during the many days I wanted to be there, but could not.

My parents, for somehow making me certain that there is nothing I cannot do, if I am willing to give the thing the effort it deserves. And for the years of proofreading, as well as summer road trips full of grammar and spelling quizzes, that made this project a little easier.

My committee, for their support and the questions that frequently seemed simple, but always provided profound contributions to the final results.

Mark Kragh, whose coding and patient analysis of results chopped many days from the completion date of this research.

I also deeply acknowledge the support and contributions my advisors, Professors Murali Tummala and John McEachen. Their willingness to accept me as a student, researcher, and eventual colleague has been continually reassuring and encouraging. Without their acceptance of the risks associated with my non-Electrical Engineering background, I would likely never have found my way to a Ph.D., or any of the (hopefully-cool) results in this work and works-to-be.

THIS PAGE INTENTIONALLY LEFT BLANK

CHAPTER 1: Introduction

Despite billion-dollar businesses dedicated to information security and thousands of proposed solutions, computer network attacks continue to increase in variety and severity. These attacks can have significant financial impact; more importantly, they threaten the foundation of the internet by enabling censorship and eroding expectations of reliability, trust, and security.

Even after decades of efforts to identify and mitigate cyber attacks and the increased resilience of cloud services, network service providers are unable to consistently defeat one of the oldest of cyber attacks: denial-of-service (DoS). Nearly 30 years after the first DoS attack, these types of attacks remain the biggest fear of network service providers and the second-biggest fear of large organizations, as illustrated in Figure 1.1 [1]. Nearly 20% of network service providers suffer, on average, more than 30 DoS attacks a day, as shown in Figure 1.2. Each DoS attack consumes resources and imposes costs (even the attacks that fail), and the cost of successful DoS attacks doubled between 2016 and 2017 [1].

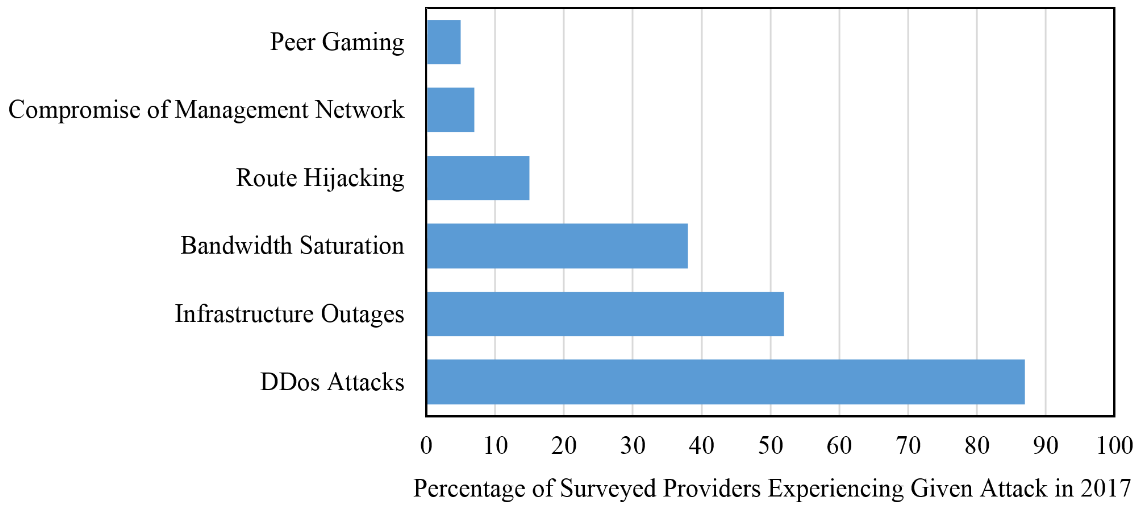


Figure 1.1. Most severe cyber threats, as perceived by network service providers in 2017. Adapted from: [1].

Limiting these costs and improving network reliability is the motivation for this work. One

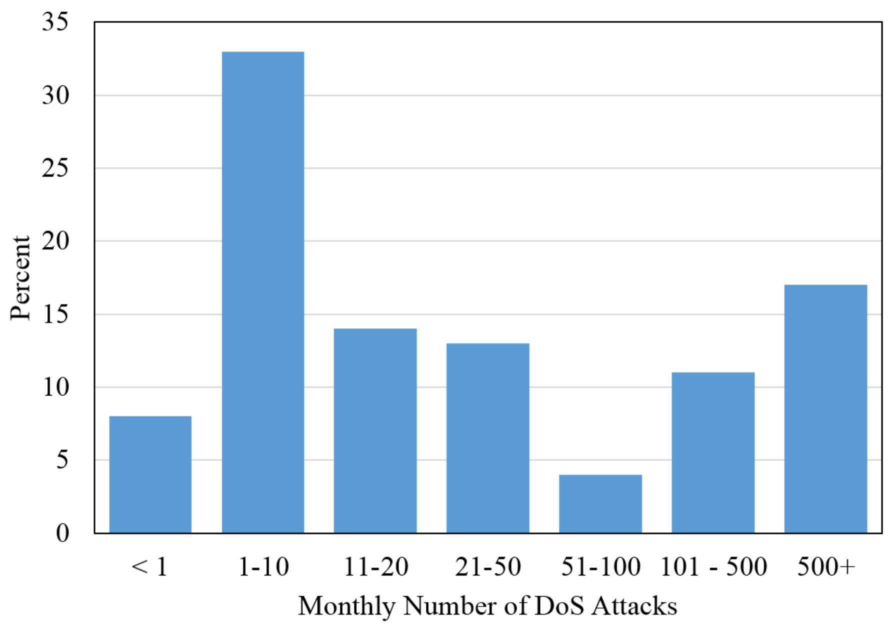


Figure 1.2. Frequency of DoS attacks observed by network service providers in 2017. Adapted from: [1].

consensus method of improving network reliability is through faster and more accurate (i.e., improved) detection of network anomalies, including DoS attacks. In order to classify a sample as “anomalous,” one must first define what is normal, or benign. So, while it is our ultimate goal to improve the status quo of network anomaly detection, we begin with improving the process of defining and modeling “benign.”

Aspects of network traffic have been known to exhibit non-Poisson and non-Gaussian behavior for some time [2]–[5]. Depending on the specific feature of network traffic that is monitored, as well as the window over which the traffic statistics are collected, the resulting distribution is frequently non-symmetric and heavy-tailed. These are characteristics that a Gaussian distribution can only approximate with error; however, Gaussian-based modeling and detection continues to be the de facto implementation used by of the statistical and machine learning anomaly detection approaches that were reviewed as part of our literature search ([6]–[8], among others).

It was demonstrated at the beginning of the decade, using a range of network traffic rates, that the α-stable distribution can provide a more accurate and flexible fit of network traffic than traditionally-used Gaussian distributions and their alternatives [9], [10]. Our review of the literature, however, identified no follow-on α-stable networking studies that harnessed this improved approach. Many *other* fields have used α-stable models to improve modeling accuracy [11], [12], though their adoption has likely been limited by the computational inefficiencies and complexities that result from the lack of closed-form solutions for (non-special case) stable distributions [13].

Beyond modeling, previous work in radar and underwater signal detection and localization (among other fields) has repeatedly demonstrated that applying stable estimation and detection algorithms in appropriate, non-Gaussian environments can significantly improve detector performance [14]–[17]. This concept has also been extended to address the impulsive noise environments frequently encountered in wireless communications [18].

Consequently, the precedents that network traffic is frequently α-stable and that stable-derived detection is more effective in non-Gaussian environments suggest approaching our goal of improved network anomaly detection through the application of stable methods.

1.1 Objective

The objective of this dissertation is the development of a computationally-tractable (i.e., scalable) method that improves the performance of a volumetric network anomaly detection system by using processes and assumptions based on α-stable distributions. To assess the accomplishment of this objective, we must construct an entirely stable-based detection system and quantify the resulting gains.

1.1.1 Design of Approach

The design concept of our approach is shown in Figure 1.3. A comprehensive review of the literature and preliminary investigation identified key detection processes that could be revised to improve overall detection system accuracy. The three processes shown in Figure 1.3 provide the framework for the intended overall approach; investigation of these areas provided the novel results in this dissertation.

The first system improvement, the focus of Chapter 3, is that the traffic model should reflect the fundamental nature of the monitored feature. This principle is commonly expressed

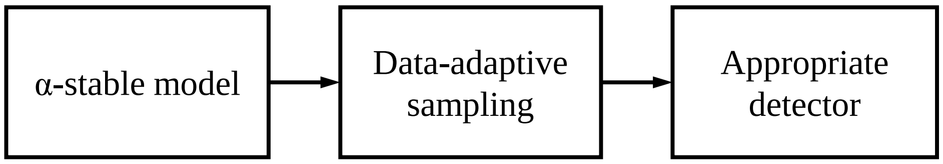


Figure 1.3. Summary of intended approach for reducing errors in statistical network anomaly detection

in the literature, but less frequently practiced. For instance, many traffic models assume Gaussian distributions of data [6], [8], but recent work and our results show that the α-stable distribution more accurately reflects many distributions of network traffic features [9], [19].

Our second process for improvement (and the focus of the novel work in Chapter 3) is adaptive data windowing. It is our prediction that, when more flexible and accurate non-Gaussian distributions are used to model the data, we can further improve overall system accuracy or response time through tuning the amount of data aggregated (e.g., window or sub-window size) prior to fitting a distributional model.

The final improvement is that the nature of the signal and noise inputs must be incorporated into the design of the *detector* [20]. To elaborate on the importance of this final improvement, the consensus of the literature is that Gaussian modeling and detection performance degrades greater-than-linearly as the environmental noise grows more impulsive and, thus, less Gaussian (i.e., the characteristic exponent α of the noise goes from $2 \rightarrow 0$) [21]–[23]. As a result, detectors using stable-derived implementations have been shown to improve detection or reducing estimation errors by two orders of magnitude in some cases [16], [17], [22], [24]. Developing non-Gaussian approaches for the detector processes is the focus of both Chapter 4 and Chapter 5.

To further focus the research and encourage novel results, we propose the following additional design restraints.

1.1.2 Implementation Restraints

The first restraint is that the proposed implementation utilize scalable techniques with real-time potential. This restraint should improve the potential for application of any successful solution while also focusing the scope of efforts (i.e., reducing the number of potential solutions).

The next restraint is that the investigation should consider adaptive techniques to improve modeling and detection accuracy. This ensures investigation of a relatively unexplored area of network anomaly detection, as will be discussed in Section 1.2.

We also restrain the system in terms of traffic perspective. The proof-of-concept system is designed with the assumption that it has an aggregate view of all traffic destined for a target, such as at the border of a large, single-homed network. It is envisioned that the developed techniques will ultimately be extensible to volumetric monitoring and detection within the internet core or even at the borders of smaller networks; however, the detection system proof-of-concept ultimately assumes that the observed traffic is the *sum total* of all traffic that could be classified as benign or attack.

The final restraint is that of the types of anomalies to be investigated. The proposed system is designed to detect, and assessed against, volumetric DoS attacks that exhaust resources at or near the target through consuming assets such as transmission bandwidth or open ports (e.g., the SYN flood attack). Numerous other anomalies such as scans and worms manifest themselves, at least partly, in a volumetric deviation. While the volumetric changes from these types of anomalies *could* conceivably be detected by the proposed system, discriminating these attacks, particularly at low volumes, requires additional classification layers that are not part of the implementation proposed in this work.

Now that the motivation and methodology to develop an improved, real-time network anomaly detection system have been described, it is appropriate to review similar and alternative approaches in the literature.

1.2 Related Work

There is relatively little precedent for non-Gaussian methods in the field of statistical network anomaly detection; our literature review identified significant work by only two research groups: Scherrer et al. [5] between 2005–2008 and Simmross-Wattenberg et al. [9], [10] between 2008–2011. Tartakovsky et al. [25]–[28] have investigated more rapid methods for changepoint detection since the early 2000s. Adaptive network anomaly detection techniques have been explored by a few groups, most notably Tartakovsky et al. [25] and Thottan and Ji [6]. In terms of analysis and handling of α-stable data, Kuruoglu [29]–[31] explored approximation methods, and Gonzalez et al. [23], [32]–[34] investigated analysis methods during the late 1990s and early 2000s.

1.2.1 Non-Gaussian Network Anomaly Detection

One of the first significant works applying partially non-Gaussian approaches to network anomaly detection was by Scherrer et al. [5]. Their approach used gamma distributions and the auto-regressive fractional integrated moving average (ARFIMA) process to detect anomalies including flash crowds and DoS attacks. They began their investigation by validating that the gamma ARFIMA method could adequately model several different datasets of network traffic as well as flash crowds and injected DoS attacks. They then measured the trend of maximum-likelihood (ML) estimates of the gamma distribution shape α and rate β parameters at various time scales. By comparing the divergence in mean quadratic distance between the multi-scale trends of actual and expected α and β, they found they could differentiate between the benign flash crowd anomaly and the DoS attack. Specific performance results of their techniques are discussed in Chapter 5.

Our overall methodology borrowed heavily from Scherrer [5]; we validated the performance of our chosen traffic model using public datasets and in a variety of traffic scenarios, then proceeded to evaluate the performance of our system (though we did not attempt to differentiate flash crowds). The details of our implementations are different in that we used non-parametric estimation and detection methods derived from the stable family of distributions. Our proof-of-concept methodology was also different. Scherrer injected Iperf [35] and Trinoo [36] DoS attacks into a proprietary physical network operating at a low volume (e.g., approximately 2,000 packets per second) [5]; in constructed real time we detected real-world attacks present in a traffic stream at a typical volume in excess of 100,000 packets per second. Scherrer did not evaluate or demonstrate real-time capabilities [5].

1.2.2 Anomaly Detection Using α-stable Traffic Models

The primary applicable prior α-stable network anomaly detection work is that of Simmross-Wattenberg et al. [9], [10]. In a similar approach to Scherrer's, Simmross-Wattenberg advanced a non-Gaussian distribution as an improved method of modeling network traffic and then examined its performance in a detection system. They first compared the performance of the α-stable model to alternatives common in the literature using various proprietary datasets collected on the local university network and a Kolmogorov-Smirnov test to evaluate fit. Establishing that the α-stable distribution provided a better model than the gamma, Gaussian, and Poisson distributions for their examined datasets. They then used an α-stable -derived generalized likelihood ratio test (GLRT) to examine the likelihood that the current traffic sample contained either a flash crowd or DoS attack vice one of a set of stored, benign α-stable reference windows. Anomalous samples were generated using the Iperf [35] and JMeter [37] tools at intensities of 10, 25, 50, and 100% of baseline normal traffic. Simmross-Wattenberg's GLRT compared the α-stable parameters of both samples to classify the traffic as attack, flash crowd, or normal and found that the α-stable methods usually performed better than both their reproduction of Scherrer's gamma-based approach and a logistic regression-based classifier.

As Simmross-Wattenberg's work was the inspiration and benchmark for our research, our overall approach was largely adapted from his group's methodology. We used their validation of the α-stable traffic model as an a priori assumption, i.e., we extended the α-stable model to additional public datasets and evaluated its modeling performance against additional distributions (e.g., the Weibull and exponential) in a more relaxed manner. Our approach diverged from their methods in terms of detector design. Though we both use ML tests, our tests are different in both the applied test statistic and the number of attributes examined. To facilitate real-time implementation, we use single-valued non-parametric estimates of representative α-stable attributes vice a four-parameter α-stable ML fit of the sample [10]. Also, our system holds and analyzes only the most recent benign prior sample, while their implementation requires pre-processing and storage of a year of benign samples [10]. We constructively establish the real-time potential of our system and also examine the effects of aggregation period on model fit error; Simmross-Wattenberg's research used a fixed window size and off-line processing [10]. Finally, while both studies examine detection performance using similar percentage changes in network traffic volume due to attacks, our proof-of-concept detects real-world attacks at a traffic rate of 1.0 gigabit per second (Gbps) vice injected attacks at a traffic rate below 0.07 Gbps [10]. Note that the performance of Simmross-Wattenberg's system, as well as that of selected peer studies, is partially summarized and compared in Chapter 5.

1.2.3 Rapid Changepoint Detection

A primary focus of Tartakovsky et al. has been improving rapid changepoint detection of network anomalies [25]–[28]. The first cited work is the most applicable to our methods and is the focus of our comparison. Tartakovsky developed an online, multi-feature method that sequentially examines received traffic, categorizing it by type of packet and size as well as certain other properties. Attacks are detected based on a change in the mean of estimators based on features selected to discriminate certain types of DoS attacks. Their solution is adaptive in a constant false alarm rate (CFAR) sense, varying the detection threshold to maintain a specified false alarm rate (FAR), with the ultimate objective of shortening the time until detection. Using a complete simulation, they compared the performance of their algorithm to that of the common exponentially-weighted moving average (EWMA) and cumulative sum (CUSUM) algorithms in terms of FAR and detection delay. Because of their significantly different objectives and methodology, their performance results are not comparable and are not discussed.

Similar to Tartakovsky's group [26], our approach is a non-parametric changepoint detection method. The remainder of our methodology differs substantially. Their use of the sample mean as an estimator of traffic features [26] may produce unreliable measurements, as explored in Chapter 3. The comparable attribute between our approaches worth identifying is that of adaptive methods. Tartakovsky's approach focused on varying the *threshold* to maintain a CFAR [26], an approach that inevitably sacrifices detections, while our exploration of adaptive methods focuses on optimizing window and sub-window size to improve absolute performance.

1.2.4 Adaptive Network Anomaly Detection

Our review of the literature identified few additional adaptive network anomaly detection techniques other than that of Thottan and Ji [6]. They developed a multi-feature GLRT detector to identify network anomalies based on changes in the correlation of the features. Specifically, their detection features included in- and out-bytes, datagrams, and acknowledgements collected at 15-s intervals. Applying a first-order auto-regressive model, they used mean, variance, and co-variance of 2.5 minute feature vectors to identify anomalous trends and subsequently test whether that trend fell outside error bounds. They evaluated their system using a physical, two-router, 7-subnet departmental network and validated performance by comparing system alerts to logs of native network faults. The adaptive methods in Thottan and Ji's approach appear to be made in terms of adapting the thresholds to maintain a low FAR, though no algorithm for adaptation is presented.

Again, our approach differs from Thottan and Ji's [6] in substantial ways that have been previously discussed. In addition to bounding the state of the practice in terms of adaptive network anomaly detection techniques, Thottan and Ji's work was interesting because they were authors of one of the first papers we reviewed that, while specifically acknowledging non-Gaussian trends in their data, categorized the incurred error that resulted from Gaussian modeling as acceptable [6]. Repeated encounters with this practice during our literature review encouraged our investigation of non-Gaussian anomaly detection methods and development of the term *avoidable errors* to describe errors incurred from applying Gaussian distributions to non-Gaussian processes.

1.2.5 Approximation of α-stable Distributions

Kuruoglu's work established much of the theoretical basis for approximating α-stable distributions using varieties of computationally-tractable alternatives including Gaussian and Lévy distributions [29], [30]. His 2003 work showed that it is possible to approximate a positive α-stable (PaS) distribution (defined by having $\alpha \in (0, 1)$) with a finite sum of Lévy distributions. His methodology is discussed in more detail in Chapter 4.

Our research initially intended to apply Kuruoglu's methods as the basis for our traffic models because of the perceived computational efficiency that would result from a finite, closed-form model. As part of this investigation, we incrementally extended Kuruoglu's work to an actual network traffic dataset, developing written and MATLAB algorithms for part of the range of the PaS distribution [30]. For reasons discussed in Chapter 4, the development of this approach stopped, however, when exploration of higher-volume traffic datasets identified that this methodology was not extensible to the majority of our datasets and would limit the applicability and flexibility of the proposed system.

1.2.6 Zero Order Statistics

Gonzalez and Arce explored stable-based estimation in the late 1990s, developing a new family of robust estimators designed specifically for heavy-tailed data [23], [32]–[34], among others. Their initial method, the myriad, uses a formula derived from the stable special-case Cauchy distribution. Their subsequent efforts developed a family of Gaussian-equivalent estimators, zero-order statistics (ZOS), which provide non-Gaussian alternatives for estimation of location, dispersion, and power.

Our literature review identified that these ZOS estimators have not been previously applied in network anomaly detection and remain largely unapplied outside of the developing research group. We utilize two ZOS estimators, specifically zero-order location (ZOL) and an adaptation of zero-order dispersion (ZOD), in our proof-of-concept detection system and demonstrate their superior performance as compared to Gaussian equivalents. We also adapted their methodology to develop our novel, heavy-tailed estimator, the Lévy location estimate (LLE) (discussed in Chapter 4). We have only completed a preliminary investigation of LLE performance, but to date it seems to be largely equivalent to their stable-derived myriad and ZOL. We have not completed evaluation of their power estimator and its potential for network anomaly detection; detailed examination of its performance as well as that of the LLE are items of future work.

1.3 Dissertation Organization

This dissertation is organized as follows. Technical topics that are foundational to our approach that would disturb the narration of our methodology and results are contained in Chapter 2.

A roadmap for the remainder of this dissertation, beginning with our exploration of the applicability of the α-stable traffic model and the adaptive techniques it enables, is given in Figure 1.3. This discussion, and a quantification of the resulting reduction in modeling error, is contained in Chapter 3.

After validating the applicability of the α-stable model to three datasets and four traffic scenarios, we derive and develop methods of both approximating and estimating α-stable distributions in Chapter 4, including our novel LLE. In this chapter, we also compare the accuracy of the developed estimators.

The design of the end-to-end detection system, proof-of-concept implementation methodology, and system performance results are discussed in Chapter 5. We show that the α-stable approach to network anomaly detection improves detection rates by 3–8% over Gaussian methodologies at a fixed FAR $\leq$ 1%. We also show that our adapted ZOS estimate of dispersion exhibits 99.96% detection rates at lower FARs than Gaussian methods. Finally, the results in this chapter constructively demonstrate the real-time potential of our implemented detection system.

The overall methodology and the results of Chapters 3–5 are summarized in Chapter 6, along with contributions and recommendations for future work.

The proof-of-concept detection system code used to obtain the results in this dissertation are contained in the Appendix.

THIS PAGE INTENTIONALLY LEFT BLANK

CHAPTER 2: Background Technical Topics

In this chapter, there are no results or arguments that are fundamental to the objective of this work; instead, technical background and implementation details that support the arguments and conclusions in Chapters 3, 4, and 5 are provided. The reader may choose to skip this chapter entirely, referencing topics based upon discussion in subsequent chapters, or skim the topics addressed here in order to ensure familiarity with our methodology.

We first review terms in this work to which we may ascribe meanings different than those in common usage or that may have varying definitions depending on the field of application. We then provide background on the α-stable distribution and the stable family as a whole, including definitions, descriptions of the four stable parameters, and special cases. Methods applicable to our modeling and estimation work, such as ML, log likelihood (LL), and distance measurements are described next. We conclude with an overview of the evaluation methodology for our detection system, defining important measures and describing the receiver operating characteristic (ROC).

2.1 Terminology

Let us begin with terminology. Certain terms used throughout this dissertation may have common or alternate meanings different from our intended purposes. We clarify their usage in this dissertation as follows:

- *sample* refers to a data window of network traffic vice probabilistic collection of data.
- *.pcap* refers to a network traffic capture computer file obtained using the Wireshark family of network monitoring tools [38].
- *trace* refers to a network traffic capture collected via any method.
- *scenario* refers to a type of traffic (e.g., attack, normal, or transition) as discussed in Chapter 3.
- *normal* refers to regular or benign (i.e., absence of an attack) traffic vice the Gaussian distribution. We will endeavor to always use the term *Gaussian* when referring to that statistical distribution of data.

- *window(s)* refers to the collection period, in seconds, of a data sample and is defined in Chapter 3.
- *attack traffic*, unless described otherwise, refers to dataset samples that contain attack packets mixed with benign background traffic. Unless otherwise specified, this term does not refer to traffic samples that exclusively consist of attack packets (i.e., samples where all benign traffic was removed).
- *lightweight* refers to a method that incurs relatively little computational cost and possesses the potential for real-time application using commodity hardware.
- *DoS* is used generally to include both non-distributed and distributed denial-of-service (DDoS) attacks.

Note that Wireshark® [38], Windows®, Python®, MATLAB® [39], and STABLE® [13] are all reserved titles of software applications. The displayed format will only be used when referring to the given software application, and these terms are reserved for their rightful owners regardless of the use of the ® symbol. Note that uppercase "Windows" refers to the software, while "window" and "windows" refer to the length in seconds of the data sample(s).

2.2 The Stable Distribution

The objective of this work is to examine modeling and detection gains that can be achieved through selective application of the α-stable distribution and associated methods vice the frequently-used Gaussian; thus, background on the stable distribution is essential to explain *why* our methodology may be advantageous.

Stable distributions were originally developed in the 1920s by Lévy and Khinchine [40] and were first popularly applied to financial analysis and forecasting in the 1960s by Mandelbrot [41]. Stable distributions have been applied to improve the accuracy of modeling random processes that exhibit non-Gaussian behavior across a significant body of work in fields as varied as finance, signal processing (including radar, sonar, and wireless noise), animal behavior, and geologic processes [11], [14], [15]. Stable methods are de facto standard for financial modeling of appropriate asset classes as well as risk assessment [42].

The family of stable distributions is also referred to in the literature as Lévy stable, Pareto stable (or Paretian), and α-stable. In this work, we use *stable* to refer to the family, *Lévy* to refer to the special case, and α-*stable* to refer to stable distributions that are not special cases. Extensive background and theory regarding stable distributions are available in [40] and [41], among others. We now review the theoretical aspects of stable distributions that are necessary to understand the applications in this work.

2.2.1 Definition

Special cases of the stable family include the Cauchy, Gaussian (or normal) distribution, and the Lévy (or Pearson V) distributions. Except for these special cases, a closed-form solution for the probability density function (PDF) of an α-stable random variable (RV) $Z \sim S(\alpha, \beta, \gamma, \mu)$ does not exist [41]; Z is instead defined through its characteristic function [40]

$$\mathbb{E}\big[e^{i\theta Z}\big] = \exp\left(-\gamma^{\alpha}|\theta|^{\alpha}\left[1 + i\beta \tan\left(\frac{\pi\alpha}{2}\right)\mathbf{sign}(\theta)\Big(|\gamma\theta|^{1-\alpha} - 1\Big)\right] + i\mu\theta\right) \tag{2.1}$$

for the case of $\alpha \neq 1$, and

$$\mathbb{E}\big[e^{i\theta Z}\big] = \exp\left(-\gamma|\theta|\left[1 + i\beta\left(\frac{2}{\pi}\right)\mathbf{sign}(\theta)\ln\big(\gamma|\theta|\big)\right] + i\mu\theta\right) \tag{2.2}$$

for $\alpha = 1$.

2.2.2 Parameters

The α-stable distribution exhibits tremendous flexibility as a model because it is described by four parameters: $\alpha, \beta, \gamma, \mu$. For the purposes of this work, the parameter values of α-stable RV Z are specified using the form $S(\alpha, \beta, \gamma, \mu)$. These parameters and some of their properties are listed in Table 2.1 and described in more detail below.

Table 2.1. Parameters of the α-stable distribution. Source: [20].

Parameter	**Property**	**Range**
α	Tail size	$(0, 2]$
β	Asymmetry	$[-1, 1]$
γ	Spread	$[0, \infty)$
μ	Location	$\Re$

The characteristic exponent α controls the tail size of the distribution and, thus, affects the height of the peak as well [41]. A very small α leads to a heavier tail and a very narrow distribution as shown in Figure 2.1.

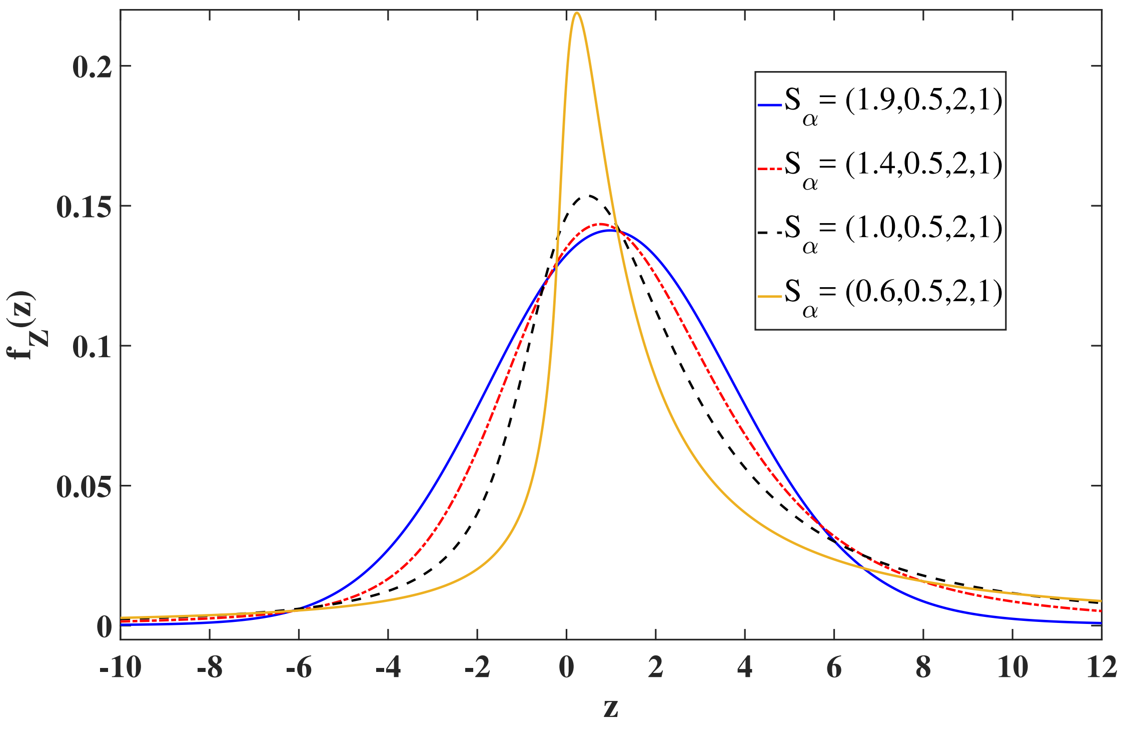


Figure 2.1. Effect of varying α on PDF peak height and tail size. Adapted from: [41].

The skew of the distribution is given by β and is reflected in the placement of the heavy (infinite) tail. Values of $\beta \in (0, 1]$ place the tail towards $+\infty$. Assigning the opposite value to β reflects the distribution across the origin. For $\beta = 0$, the distribution is symmetric, such as for the Cauchy or Gaussian distributions. (For the Gaussian case convention assigns a value of zero, though technically β is undefined [40]).

The scale parameter γ specifies the *dispersion* of the distribution, which is roughly equivalent to the Gaussian concept of variance [41]. Large values of γ result in a wider distribution with a lower rate of change around the mode.

The shift parameter μ influences the placement of the distribution; it is roughly equivalent to Gaussian mean. The shift parameter μ does not specify the mode of the distribution in the zero parameterization; its location must be numerically determined [41].

These parameters can fundamentally alter the shape and support of the distribution, permitting very expressive data models. Some representative stable PDFs for various parameter combinations are shown in the probability density plot of Figure 2.2.

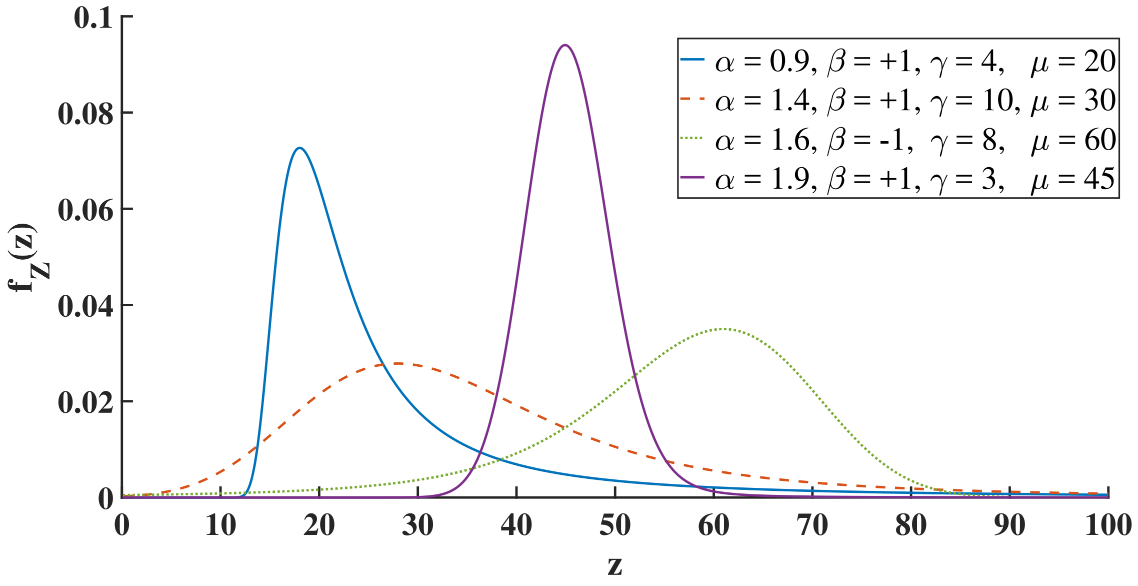


Figure 2.2. Distribution of Z for varying values of its parameters

The function values of a stable distribution are affected by the *parameterization* used to define the characteristic functions in (2.1) and (2.2). Unless otherwise specified, all figures and results in this work were determined using the zero parameterization as defined by Nolan [41]. When a stable distribution is defined in the manner given above, μ and γ can be considered shift and scale parameters [41] such that

$$\frac{Z - \mu}{\gamma} \sim S(\alpha, \beta, 1, 0). \tag{2.3}$$

The zero parameterization is used by MATLAB, as all parameters are continuously-defined [41]. The other implications of the various parameterizations historically used to define stable distributions and their effects on the values of the distribution are beyond the scope of this work; the interested reader is referred to [40] and [41] as well as their original sources.

Let us now discuss the special cases of the stable distribution, as all of these cases are utilized in this work.

2.2.3 Special Cases of the Stable Distribution

As previously discussed, for certain values of the characteristic exponent α, the stable distribution can be defined using a closed-form solution; these are the *special cases* of the stable family.

The Lévy special case corresponds to $\alpha = 0.5$, is wholly-skewed (i.e., one-sided), and has the heaviest tail of the special cases. The Lévy RV Z is distributed per

$$f_Z(z) = \sqrt{\frac{\gamma_Z}{2\pi}} \frac{1}{(z-\mu_Z)^{3/2}} e^{-\gamma_Z/2(z-\mu_Z)}. \tag{2.4}$$

The Cauchy special case corresponds to $\alpha = 1$, where the distribution is defined as

$$f_Z(z) = \frac{\gamma_Z}{\pi}\left(\frac{1}{(z-\mu_Z)^2 + \gamma_Z^2}\right). \tag{2.5}$$

Finally, if $\alpha = 2$, the Gaussian special case applies, and the distribution is defined as

$$f_Z(z) = \frac{1}{\sqrt{2\pi\sigma_Z}} e^{(z-\mu_Z)^2/2\sigma_Z^2}. \tag{2.6}$$

Note that the closed-form σ_Z in (2.6) does not correspond to the stable characteristic function parameter value of γ_Z [40].

The PDFs of these special cases are shown in Figure 2.3, for $\gamma_Z = \sigma_Z^2 = 1$ and $\mu_Z = 0$. Notice the differences in tail size between the special cases; the implications are significant and are examined in more detail in Chapter 4. In short, the Gaussian has an exponential tail that decays much more rapidly than the Cauchy and Lévy cases, which possess an α-stable power-law (or heavy) tail.

The Lévy and Cauchy cases, in particular the Cauchy, have been repeatedly used to provide improved, non-Gaussian solutions to modeling and detection problems, particularly in wireless communications, radar, and underwater detection problems [17], [18], [43].

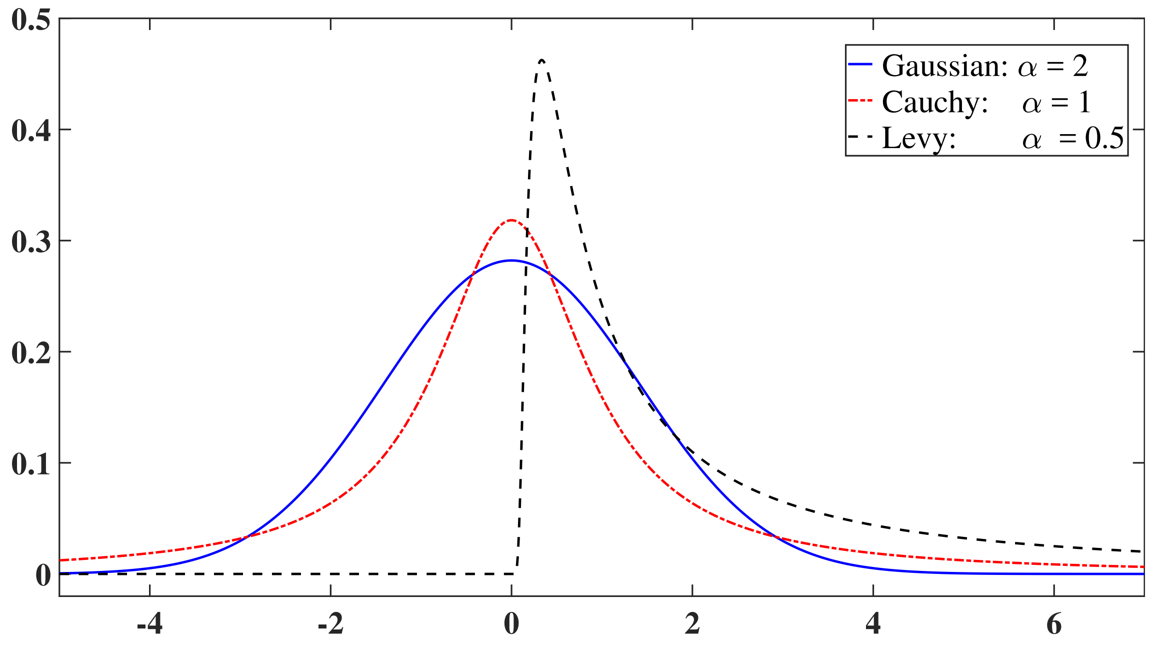


Figure 2.3. Comparison of form and tail of PDFs of the three special case stable distributions. Adapted from: [41].

.

2.2.4 Constraints of Stable Distributions

In this section, we elaborate on the stable properties and constraints already discussed in Section 2.2.2. Two constraints are of particular concern:

1. Summation constraint: A simple formula for the sum of two α-stable RVs X and Y *only* exists when the characteristic exponents of the summands are equal, i.e., $\alpha_X = \alpha_Y$ [41].
2. Constraint on higher order moments: Only moments of order p such that $p < \alpha$ are finite [41].

The summation constraint is pertinent to our work because network traffic is the sum of many stable-distributed processes characterized by many different values of α. There is, thus, no simple way to mathematically deconstruct streams of traffic from a single, monitored α-stable trace.

During the data analysis performed for this dissertation, we have observed $\alpha \in [0.3, 2]$ depending on aggregation period and network traffic dataset. The implication of the moment constraint, combined with our observations, is that traditional first and second-order moments from Gaussian processes (i.e., mean for $p = 1$ and power for $p = 2$) cannot be assumed by default to exist for α-stable processes (to use, for instance, in a detection system).

Given these definitions and constraints, we now describe our approach for modeling network traffic using the stable distribution.

2.3 Modeling the Data: Maximum Likelihood Estimation and Log Likelihood

Our selected approach, maximum likelihood estimation (MLE), has a venerable history in applied signal processing and is a-parametric in that the principles can be applied to any data set once a distribution model is chosen a priori. Additionally, the asymptotic efficiency of the MLE approaches optimum for sufficiently large data records [44].

Other estimators that may be more accurate in certain cases, specifically minimum variance, cannot be applied to our model in all cases because of the moment-order constraint (MOC) (per Section 2.2.4) and because the minimization techniques require assumptions regarding the distribution of residual errors that may not be appropriate. These constraints are typical of minimum variance estimation, and as a result, most implemented estimators use the ML method [44].

2.3.1 MLE

Let us now examine MLE as applied in this research. The MLE method maximizes the fit of a chosen distribution to a set of probabilistic data d_i.

For the purposes of this dissertation, assume d_i is an independent random sample of data generated by α-stable process Z over a finite period of time, the data window ϖ_i.

The random variable Z can be characterized as a stably-distributed RV, parameterized such that

$$Z \sim f_Z\Big(z; \boldsymbol{\theta}_Z\Big) \sim S(\alpha_Z, \beta_Z, \gamma_Z, \mu_Z), \tag{2.7}$$

where $\boldsymbol{\theta}_Z$ are the unknown parameters $(\alpha, \beta, \gamma, \mu)$ of $f(z)$.

The probability of repeated observations of z for $i \in [1, N]$ given $\boldsymbol{\theta}_Z$ is then given by

$$p(Z) = \prod_{i=1}^{N} f_Z(z_i; \boldsymbol{\theta}_Z). \tag{2.8}$$

Now assume we wish to estimate the unknown $\boldsymbol{\theta}_Z$ using the known sample d_i, equivalent to our repeated observations of z.

Over the range of potential values of $\boldsymbol{\theta}_Z$, the likelihood of obtaining the sample d_i for a specific $\boldsymbol{\theta}_Z$ is given by rewriting (2.8) as the likelihood

$$L(Z; \boldsymbol{\theta}) = \prod_{i=1}^{N} f_Z(\boldsymbol{\theta}_Z; d_i). \tag{2.9}$$

Our MLE of Z can then be defined as the values of $\boldsymbol{\theta}$ that maximize (2.9) [44], or

$$\hat{\boldsymbol{\theta}}_{ML} = \arg\max_{\boldsymbol{\theta}} \prod_{i=1}^{N} f_Z(\boldsymbol{\theta}_Z; d_i) \tag{2.10}$$

To avoid exceeding floating point constraints, without changing the result we can take the natural logarithm of (2.10) to obtain the LL MLE per

$$\hat{\boldsymbol{\theta}}_{ML} = \arg\max_{\boldsymbol{\theta}} \sum_{i=1}^{N} \ln f_Z((\boldsymbol{\theta}_Z; d_i). \tag{2.11}$$

We use both forms of MLE in this work, though generally we use (2.10) for theory and apply (2.11) in our algorithms.

In summary, for specific data window ϖ_i, $\hat{\boldsymbol{\theta}}_{\boldsymbol{ML}} = \left(\hat{\alpha}_Z, \hat{\beta}_Z, \hat{\gamma}_Z, \hat{\mu}_Z\right)$ represents the best estimated stable fit to the data d_i. The MLE stable parameter vector $\hat{\boldsymbol{\theta}}_{\boldsymbol{ML}}$ is determined by MATLAB as part of the fitting and estimation routines of the *distributionFitter* tool [39]. This is the tool that we used to produce many of our ML distribution fits, or “best fits.”

STABLE provides an alternative, proprietary ML routine for fitting stable distributions to data [13]. Based on the maturity of STABLE, this ML routine was used to produce all results that exclusively use parametric stable fits (such as trends in α-stable parameters) and the majority of results in Chapter 4 and Chapter 5. The MATLAB ML routines and estimates were used to produce the results in Chapter 3, which compare the modeling performance of various distributions as well as most figures (for ease-of-use purposes). MATLAB routines were also used to evaluate the fits of Lévy mixture approximation (LMA) methods, as these results are not central to the estimation and detection contributions of this work.

Empirically, the MATLAB stable tools provide similar results to those obtained from STABLE. The one noted exception is that when α approaches its limits of $(0, 2]$. Convergence of numerical estimation routines is a known issue that is well-documented in the literature for values of $\alpha < 0.4$ [13], [45], and this can lead to MATLAB and STABLE ML estimates of $\boldsymbol{\theta}_Z$ that differ non-trivially. Some minor convergence differences also appear to occur at α values very close to 2, likely due to accuracy thresholds within the algorithms, and these differences usually only apply to the returned value of β. Given these statements, however, it is important to note that nearly all data samples across all datasets fell in a range of $\alpha \in [0.4, 2]$; as such, these convergence issues did not substantially affect our work or results.

The MLE method described above is used when a sample is known (or assumed) to follow a distribution a priori. When faced with a sample that originates from a random process following an unknown distribution, ML estimates of best fit can be obtained using multiple candidate distributions. The LL can be used to quantify which candidate distribution is *most likely* to have produced a given sample.

2.3.2 Log Likelihood

The LL value can be obtained after using the MLE algorithm to estimate the unknown parameters $\boldsymbol{\theta}$ of a distribution. This is the "best fit," or the specific parameter values that are most likely to have produced a known data series $\boldsymbol{z}$.

The likelihood function of the data $\varphi(\hat{\theta})$, given these estimated parameters, can be defined in terms of the product of the probability of observing each individual value in $\boldsymbol{z}$ for $\hat{\boldsymbol{\theta}}$ such that

$$\varphi(\hat{\boldsymbol{\theta}}) = f(z_1; \hat{\boldsymbol{\theta}}) \times \ldots \times f(z_n; \hat{\boldsymbol{\theta}}) = \prod_{i=1}^{M} f(z; \hat{\boldsymbol{\theta}}) \tag{2.12}$$

for integer $i \in [1, M]$ [46].

Note that the final term of (2.12) is very similar to our method for determining $\hat{\boldsymbol{\theta}}$ in (2.8). By taking the natural logarithm of both sides of (2.12), we obtain the LL of our ML-based best fit [46] as given by

$$\ln \varphi(\hat{\theta}) = \sum_{i=1}^{M} \ln f(z; \hat{\theta}) \tag{2.13}$$

where z can be seen to be equivalent to our data sample d_i.

Our work in Section 3.4 uses multiple traffic scenarios from multiple network datasets to determine the most appropriate distributional model for each dataset; these scenarios and datasets are discussed in detail in Chapter 3. We processed these samples using native MATLAB functions to fit various distributions to each filtered trace and used the LL value of the ML MATLAB fit to compare fits for each scenario and dataset.

Our method was adopted from the literature, which frequently uses LL to compare various ML best fits and determine which distribution provides the *most accurate* fit to a sample originating from an unknown distribution [46]; however, the nature of the LL prevents it from being a preferred metric for comparing distributional fit across samples of different sizes because the LL can be seen in (2.13) to be the result of a sum and, as such, is dependent on sample size. An alternative to the LL is, thus, required to enable this type of comparison.

2.4 Measuring Model Fit

We decided to compare fits, or determine fit error, by measuring the distance between the sample histogram and the best fit model of the sample (i.e., best fit distribution to the sample).

The literature contains many methods for calculating this distance (i.e., similarity, dissimilarity, divergence) between data and the closest statistical approximation [47], [48].

For comparison and assessment purposes, we initially used three methods to measure this distance, and we now briefly review their formulae and important considerations for our applications. An interested reader is referred to the cited references for more in-depth derivation and analysis.

2.4.1 Kullback-Leibler Divergence

The Kullback-Leibler divergence is an entropy-based measure that is frequently used to quantify the divergence between one probability distribution (e.g., the ML fit to a data sample) and a second, expected distribution [47]. Its formula [47] is given by

$$d_{KL}\left(f_Z(z), \hat{f}_Z(z)\right) = \sum_{j=1}^{M} f_Z(z_j) \ln \frac{f_Z(z_j)}{\hat{f}_Z(z_j)}, \tag{2.14}$$

where for our purposes $f_Z(z_j)$ and $\hat{f}_Z(z_j)$ are the reference distribution and ML approximation, respectively, sampled at points $j \in [1, M]$ that correspond to the mid-points of the sample histogram bins.

2.4.2 Hellinger Metric

A frequently-used alternative to the Kullback-Leibler divergence in information-theoretic approaches is the Hellinger metric [47]. We define the average Hellinger distance [47] as

$$d_{Hel}\left(f_Z(z), \hat{f}_Z(z)\right) = \left(\frac{2}{M} \sum_{j=1}^{M} \left(\sqrt{f_Z(z_j)} - \sqrt{\hat{f}_Z(z_j)}\right)^2\right)^{1/2}. \tag{2.15}$$

2.4.3 Canberra Distance

Our final distance measure is the Canberra distance, more frequently used in clustering and machine learning applications [47]. The Canberra distance was chosen to provide diversity and enable comparison of trends from (2.15) and (2.14) relative to an independent third measure, adhering to recommended best practices [47]. The Canberra distance [47] is defined as

$$d_{Can}\Big(f_Z(z), \hat{f}_Z(z)\Big) = \sum_{j=1}^{M} \frac{\big|f_Z(z_j) - \hat{f}_Z(z_j)\big|}{f_Z(z_j) + \hat{f}_Z(z_j)}. \tag{2.16}$$

2.4.4 Preferred Distance Measure

After comparing the measures quantitatively for consistency, as well as comparing the quantitative measurements to the visual fits, we chose to quantify the fit results in this dissertation using the average Hellinger distance. We use *average* Hellinger distance to measure of the *absolute* error of fit because (2.15) does not generate positive and negative terms that can offset accumulated error during summation, as compared to Kullback-Leibler divergence. Our formula in (2.15) is a normalized version of the standard Hellinger metric to facilitate comparison between scenarios.

Now that the methodology for evaluating and quantifying the performance of our α-stable network traffic model has been established, let us describe our methods for evaluating the performance of our detection system.

2.5 Measuring System Performance

The proposed end-to-end detection implementation begins with traffic samples that are extracted from network traffic capture files, or .pcaps, and finishes with an anomaly *score*, as determined by the test statistic. After extracting thousands of both benign and attack traffic samples from the chosen datasets, the samples are applied to test statistics, and the cumulative system performance is evaluated using a ROC curve, or more typically, ROC.

2.5.1 Receiver Operating Characteristic (ROC)

A ROC is a commonly-used evaluation method for binary classification (e.g., detection) systems; it summarizes system performance for a set of data by showing the trade-off between *true positives* and *false positives* as a function of a given threshold value [49].

For a given population of attack N_A and benign N_B samples that are labeled appropriately, we can define n_{tp} as the number of attack samples that the detector properly classifies, and n_{fp} as the number of benign samples that are improperly classified as attack samples. We can then determine the true positive rate (TPR) P_d from the results [50] using

$$P_d = \frac{n_{tp}}{N_A}, \tag{2.17}$$

where the numerator and denominator are based on counts of the labeled input data.

Similarly, the false positive rate (FPR) P_{fa} is defined as

$$P_{fa} = \frac{n_{fp}}{N_B}. \tag{2.18}$$

These equations only give one point on the curve for each detection threshold τ, where P_d is plotted on the vertical axis and the FPR P_{fa} is plotted on the horizontal. To develop a full curve, the detection threshold must be varied over the range of the input data, which permits plotting the curve as a series of points given by $(P_{d,i}, P_{fa,i})$ for all evaluated thresholds τ_i. An example ROC developed in this manner is shown in Figure 2.4.

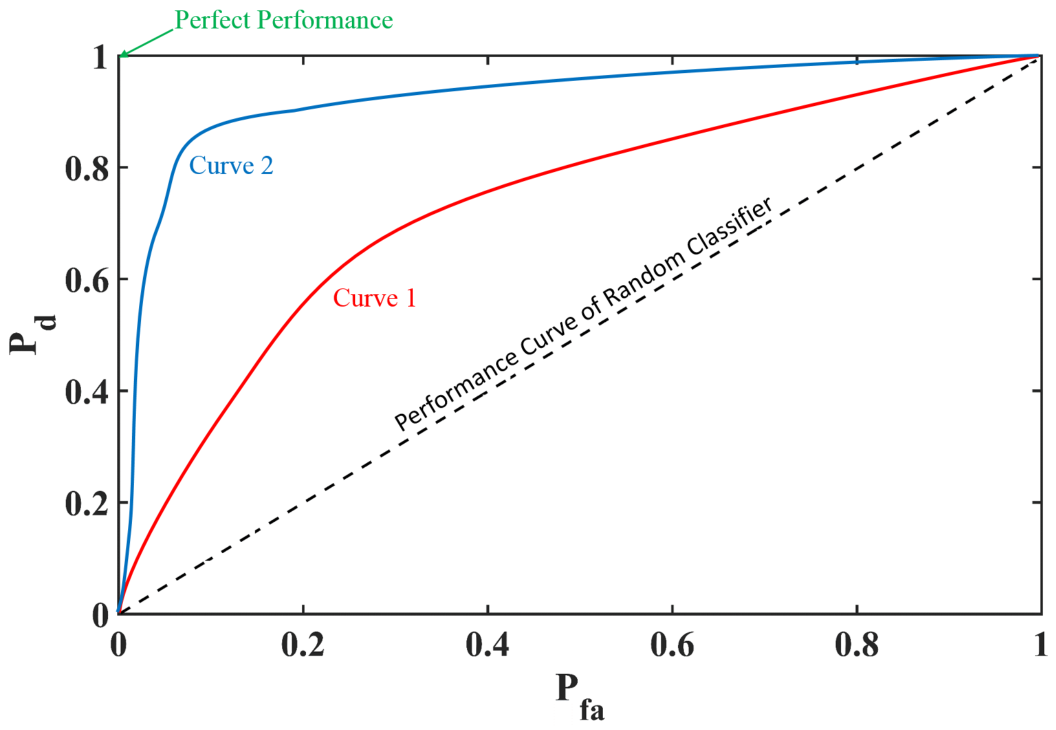


Figure 2.4. Example ROC and curves

The resulting shape of the plot provides a comparison to a random classification detector, given by the dashed curve between points $(0, 0)$ and $(1, 1)$. A detection system that plots below this straight line has worse performance than a random classifier, while the best detector is generally given by the curve closest to the point (0, 1) in the upper left of the plot.

The ROC is typically used to compare two or more detectors or approaches (e.g., Curves 1 and 2 in Figure 2.4). The area under the curve (AUC) metric is also used to provide a condensed measure of detector performance and allows quantitative comparison to other schemes. AUC is determined by numerically determining the area under a given ROC curve. The detector with the largest AUC is often considered to be the best, but the assessment of "best" is also frequently determined by implementation objectives such as the highest performance for a maximum P_{fa}(as is seen in CFAR detection implementations).

Now that we have described how to evaluate the results of a detection system, we conclude this section with a description of our methodology for obtaining detection results from our end-to-end design and input data samples.

2.5.2 Monte Carlo Simulation

Given input data samples, or for our implementation *estimates* of these data samples, our detector compares the current sample to a prior reference sample and measures the resulting change. This is done using a specific estimator and test statistic and is discussed in Chapter 4. The measured change is compared to a threshold τ and an attack "declared" if the change exceeds τ.

If we wish to robustly assess the performance of our detection system, a problem arises with our approach due to the limited set of input samples. The shortage of real-life, labeled, and recent datasets for network anomaly detector testing is a widely-acknowledged difficulty [50], [51]. This is one of the primary reasons we selected the specific datasets described in Chapter 3. Even our primary dataset has drawbacks consistent with the literature: traces constrained to a maximum 15 minutes of length and attack and benign traffic periods of uncertain duration. To make our evaluation as robust as possible given our input data constraints (which are detailed in Chapter 5), we obtained our results using the Monte Carlo method of probabilistic evaluation.

Monte Carlo analysis has a proven history of performance in areas where solutions are difficult to evaluate deterministically [52]. We believe that evaluation of detection systems based on non-closed form α-stable models is one such area. We now describe one of our typical Monte Carlo implementations, as illustrated in Figure 2.5.

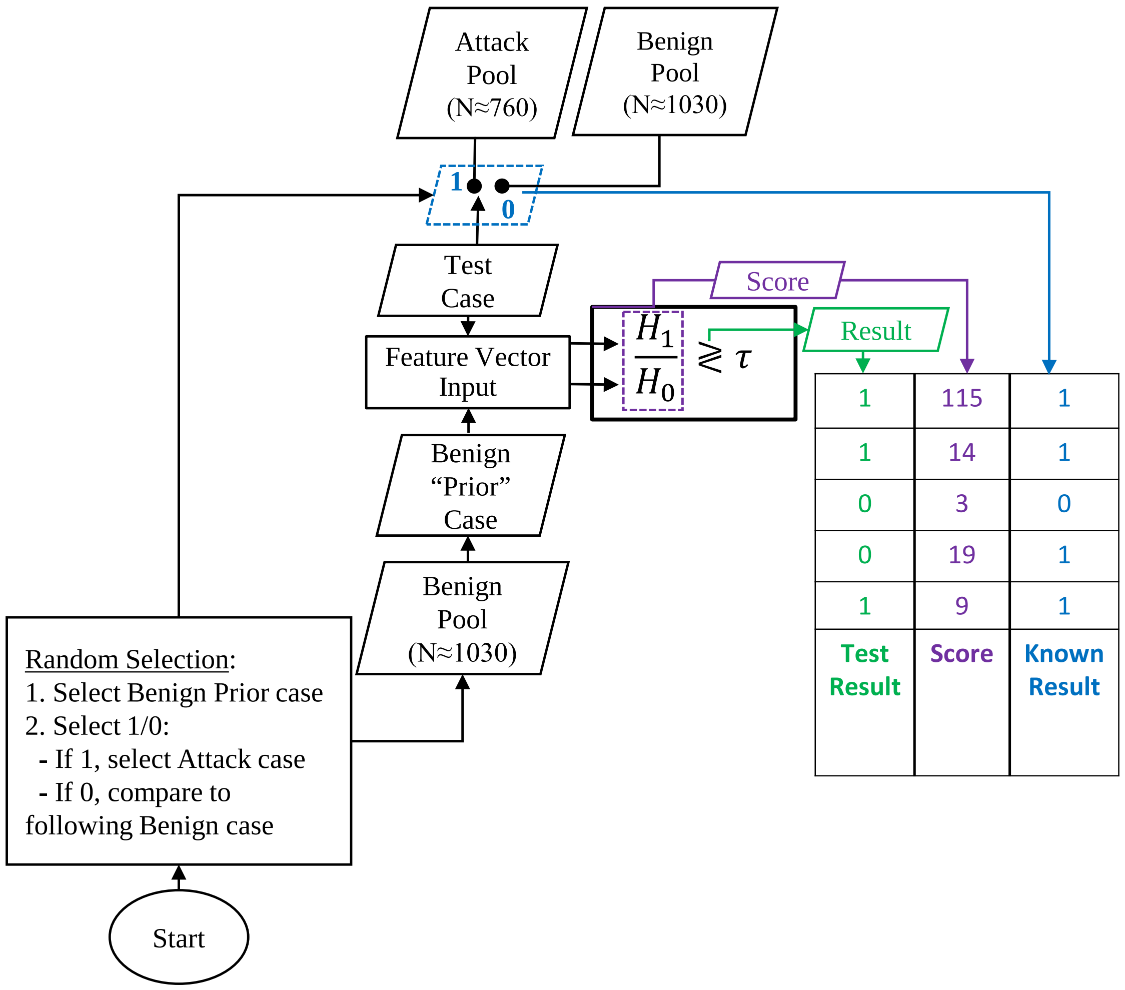


Figure 2.5. Illustration of the Monte Carlo approach for obtaining results

Our method uses data pools of labeled attack and benign samples; for our purposes, attack samples are labeled as 1 and benign samples are 0. Additionally, each sample is labeled in its sequential order of collection.

For T iterations, the algorithm randomly selects one of the benign samples (the *benign prior reference case*) and a 1 or 0 to determine the type of scenario. If "attack" is chosen, a random attack sample is selected; else the benign sample subsequent to the reference is selected. Both are then sent to the detector, which compares them using a pre-assigned test statistic from Chapter 5.

Our Monte Carlo algorithm outputs a results record consisting of T rows and two columns. The columns contain the output of the test statistic and label for the randomly-selected scenario. This record can then be evaluated by MATLAB algorithms to develop a ROC for the given test statistic, as described in the previous section.

2.6 Summary

A review of foundational technical topics that support the research described in this work was given in this chapter. These topics include the stable distribution, used for our non-Gaussian traffic model. MLE+, and measurement techniques such as LL and the Hellinger metric, are used to fit the α-stable distribution to histograms of network traffic features and measure the resulting divergence between the model and data. Finally, the end-to-end system performance measurement tools of Monte Carlo analysis and ROCs were reviewed. The remainder of this work focuses on novel methodologies and results.

THIS PAGE INTENTIONALLY LEFT BLANK

CHAPTER 3: Network Traffic Modeling

The overall approach of the next chapters follows that of our error reduction framework from Chapter 1. An enhanced version of this framework, developed during our search of the anomaly detection literature, is shown in Figure 3.1 and includes the interactions and expected outcomes of our proposed α-stable -based process improvements.

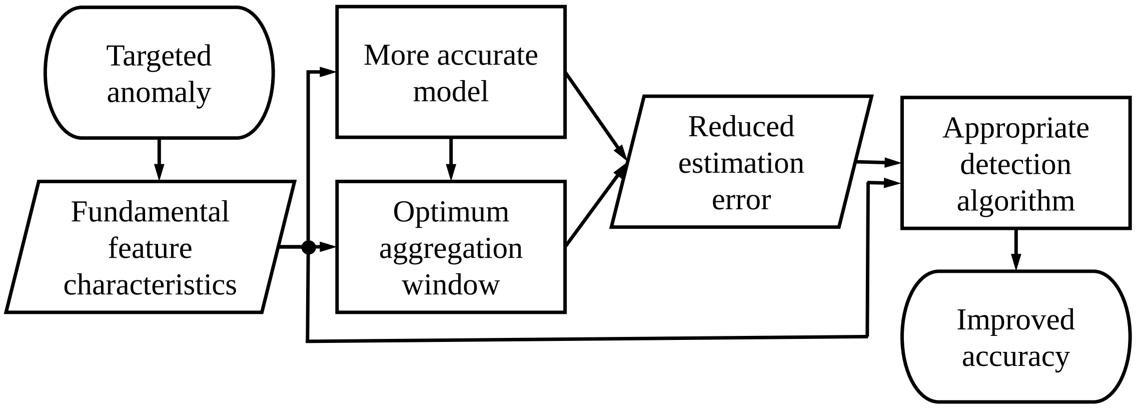

Targeted anomaly
Fundamental feature characteristics
More accurate model
Optimum aggregation window
Reduced estimation error
Appropriate detection algorithm
Improved accuracy

Figure 3.1. Model of detection processes and interactions affecting system accuracy. Adapted from: [20].

The cascading nature of avoidable errors is illustrated in Figure 3.1; these interactions suggest that small improvements in initial processes could yield significant results [20]. Investigation of these α-stable process innovations and their interactions is the subject of the remainder of this dissertation. In this chapter, we examine the four elements in the left half of Figure 3.1, while estimation error and detection algorithms are explored in Chapter 4 and the cumulative accuracy improvements are the subject of Chapter 5.

Specifically, the discussion in this chapter extends previous work [9], [10] by examining the α-stable model against a range of publicly-available network traffic datasets and traffic scenarios. After reviewing the prior work, we examine our three examined datasets for suitability and demonstrate that the traffic in each follows an α-stable vice Gaussian (or other) model. The α-stable model is shown to apply most frequently and independent of aggregation period or link conditions (e.g., benign background, attack, or high-volume background processes). Finally, the results demonstrate that use of an α-stable model, as opposed to the next-best performing Gaussian, permits optimizing sub-window and window size to reduce system response time without sacrificing accuracy.

3.1 Non-Gaussian Traffic Models: Prior Work

The Poisson and Gaussian distributions were among the first used to model network traffic, and their limitations have long been identified [2], [3], [14]. Common features of network traffic such as byte and packet count, as well as connection time and delays, have been previously shown to be asymmetric and heavy-tailed [3]. Additionally, α-stable models have been used repeatedly to improve models of the wireless networking environment, including noise and delay time [18], [24], [53].

The works cited in the previous paragraph are part of the long history of non-Gaussian, non-Poisson network traffic models that have been examined since the mid-1990s. Non-Gaussian network anomaly detection, on the other hand, has relatively limited exploration in the literature. Studies frequently utilize the Gaussian model for simplicity or speed, at times noting the non-Gaussian reality while accepting these avoidable errors [6].

3.1.1 Non-Gaussian Methods Using the Gamma Distribution

The gamma distribution appears to have been first applied in a detection system in 2007, where a gamma-ARFIMA model was applied in a change-point detector to differentiate between flash crowds and two DoS attacks [5]. The gamma distribution was shown to provide a good fit to the marginal distribution of packet rate, but the overall detection system performance, as recorded in Table 3.1, varies dramatically depending on the volume and type of the attack. Additionally, the accuracy was low in some scenarios, even for the relatively high fixed P_{fa} of 10 and 20%.

Note that each case in Figure 3.1 differs from the other attacks in terms of volume or attack packet size. Only selected cases are displayed in this table; for additional cases or details regarding the differences between attacks, the reader is referred to the source [5].

Table 3.1. Performance results from selected cases of a gamma-derived DoS detection system. Adapted from: [5].

Attack Tool	Case	Attack Intensity [%]	P_d at P_{fa} = 10%	P_d at P_{fa} = 20%
Trinoo	tT	82.9	82	82
Trinoo	tN	15.2	54	54
Iperf	R	33.8	91	93
Iperf	V	39.3	18	40
Iperf	III	21.5	48	58

3.1.2 Non-Gaussian Methods Using the α-stable Distribution

The first (and only) α-stable -based network traffic model appears in the literature in 2008 [9] and was then used in a detection system in 2011 [10]. This work applied a GLRT-based algorithm that detected DoS attacks by comparing the current window to a stored, historical reference sample from the previous year [10].

As part of their detection system, Simmross-Wattenberg et al. examined benign network traces at volumes between 0.034 and 66 Mbps collected at their university (and not publicly-available) [9]. Our work broadens their validation of the α-stable traffic model to include multiple types of traffic (*e.g.,* benign, noisy, and attack) from three different, public datasets. We first discuss the examined datasets and traffic environments, then proceed to our results.

3.2 Datasets

Three unique sources of data were examined in this work. Detailed discussion of dataset attributes is contained in the corresponding sub-section, and their individual characteristics are summarized in Table 3.2. Different portions of each dataset were analyzed and classified into one of four potential traffic *scenarios*: attack, benign, transition, and noisy. A visual example of the packet rate fluctuations corresponding to these scenarios is shown in Figure 3.2, which was developed by measuring packets per sub-window over time for the beginning of the Measurement and Analysis on the Wide Internet (MAWI) 20151115 trace. The terms *attack*, *benign*, *transition*, and *noisy* are used throughout the remainder of this work to describe equivalent scenarios. If pertinent to the discussion, the percentage of attack traffic in a transition scenario is stated.

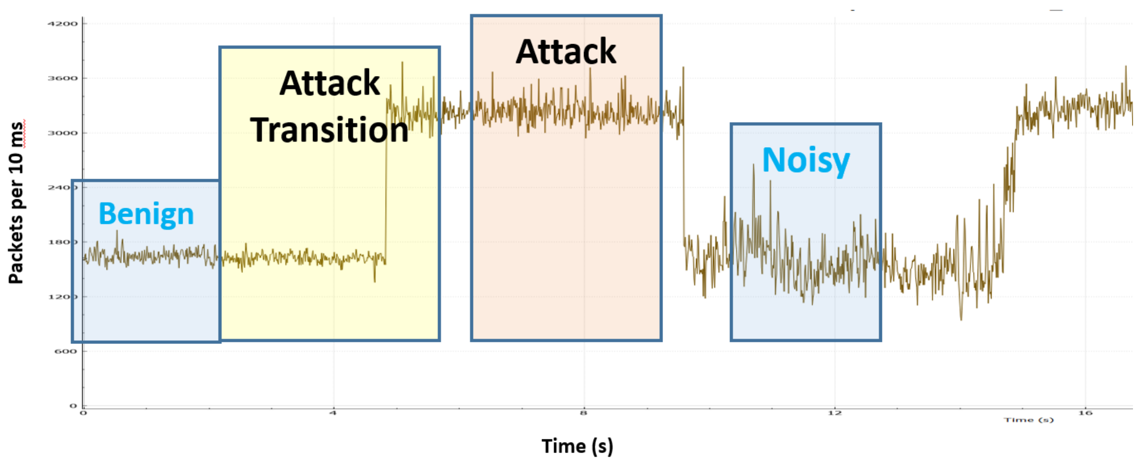


Figure 3.2. Visual characterization of four different network traffic scenarios based on packet volume

Note that our analysis only classified traffic samples as attack if labeled as such by the dataset originator or if confirmed by analysis in Wireshark. We now discuss key characteristics of the three datasets.

3.2.1 Primary Data Source: MAWI

The MAWI archive (or dataset) consists of numerous 15-minute .pcap traces collected once per day from a backbone link operating at a rate of approximately one Gbps [54]. Each trace is typically 8–10 gigabytes in size and includes anonymized, truncated copies of each packet crossing the trans-Pacific backbone link between Japan and the United States between 14:00 and 14:15 Greenwich Mean Time (GMT). After collection, each trace was subsequently processed by an ensemble of four different machine learning algorithms; the MAWI algorithms produced graphical and written summaries of major detected anomalies [54].

Since the MAWI connection serves as a real-world backbone link, MAWI traces are one of the most realistic, publicly-available data sources. The partial labeling of the archived traces assists in identifying interesting anomalies for subsequent in-depth analysis. The traffic mix itself contains varying percentages of benign background traffic, attacks including multiple varieties of DoS, and automated process traffic (e.g., scans). Because of the link characteristics and variety of traffic, we determined that MAWI data was the most realistic and, thus, “best” data source for validating our modeling and detection methods.

To select appropriate traces for both purposes, we began by examining the trace summaries for possible DoS attacks of appropriate volume and duration. We then used Wireshark to examine packet headers in the candidate traces, positively classifying high-volume flows as anomalous based on common attack indicators such as Transmission Control Protocol (TCP) flags; packet size, internet protocol (IP) address, or port distributions; and bi-directional flow characteristics.

This analysis identified three specific traces of interest for our work, .pcaps from 14 and 15 November 2015 and 28 April 2016. These traces are referred to as 20151114, 20151115, and 20160428, respectively. The 20151114 and 20151115 traces contain benign periods with a periodic, high-volume DoS attack, as can be seen in Figure 3.2. For our purposes, these are *high volume* attacks, as the additional attack traffic volume roughly equals the benign traffic volume observed during the OFF periods of the attack.

The 20160428 trace was selected to test the ability of our detection system to identify a *low volume* attack; the magnitude of this anomaly varies between 15–30% of benign background and can be seen in Figure 5.4. As detailed discussion of the attack specifics is more appropriate for Chapter 5, we now briefly describe the other datasets we used to extend the prior work in α-stable traffic modeling and validate our approach.

3.2.2 ISCX

The Information Security Center of Excellence (ISCX) dataset was produced by the University of New Brunswick and contains whole-day .pcaps of artificially-generated traffic on a physical network [55]. Two days of traffic from this dataset were analyzed. The 14 June dataset contains only benign network traffic produced by automated agents running on the hosts in the physical test-bed network shown in Figure 3.3. During the 15 June dataset, two types of DoS attacks are launched from compromised hosts on the network. The high-volume attack is a typical SYN-flood, while the low-volume attack uses the *slowloris* script [56].

The ISCX trace was minimally analyzed and not used at all in our detection work due to the small size and relatively low packet volume of the network. This small link rate required

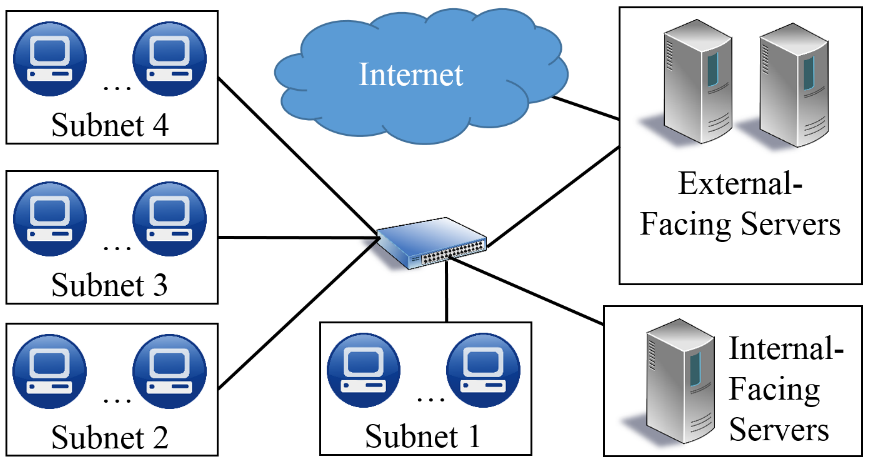


Figure 3.3. Network topology used to create the ISCX dataset. Adapted from: [56].

large data windows (on the order of 300 seconds) and still produced many zero-packet count sub-windows that made modeling the benign and attack traffic difficult, regardless of distribution.

3.2.3 MACCDC

The Mid-Atlantic Collegiate Cyber-Defense Competition (MACCDC) dataset contains .pcap captures of all traffic on the network during a network defense competition [57]. During this competition, competitors must upgrade and maintain vital network services on a business-style local network containing 10-20 hosts and servers, while an independent red team attempts to infiltrate and disable their network. As such, the dataset contains semi-realistic background network traffic as well as a large amount of reconnaissance and exploitation traffic.

A typical local network "spoke" of the overall competition network is shown in Figure 3.4. Based on our analysis, the datasets appear to be obtained from the "hub" router that connects each spoke and the red team.

The MACCDC traces were also minimally analyzed, primarily because the large volume of malicious traffic appeared to create traffic distributions that were very different from the distributions available in the ISCX and MAWI traces; the MACCDC packet volume histograms were very flat as compared to those obtained from the other analyzed datasets

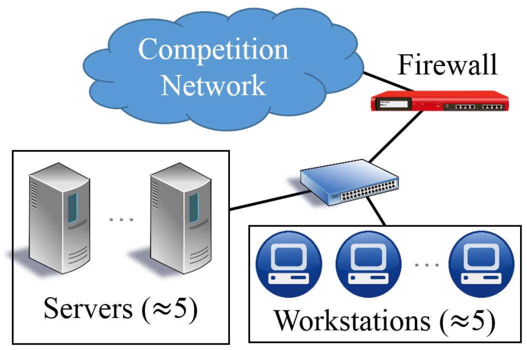


Figure 3.4. MACCDC dataset local network topology. Adapted from: [58].

and also different from benchmarks in the literature [5], [9]. Furthermore, the traces were unlabeled and relatively short in terms of time, frequently varying between eight and 20 minutes, which made it difficult to classify and separate attack and benign trace portions into usable numbers of samples for our detection system.

Accordingly, after initial analysis both the MACCDC and ISCX datasets were used to demonstrate the versatility of the α-stable distribution as a traffic model rather than as data sources for our detection system. The real-life traffic and high link rate characteristics of the MAWI datasets made these traces our choice to validate the detection system described in Chapter 5.

Note that "Average Volume" in Table 3.2 was calculated by dividing the total trace size, in packets, by the trace duration, in seconds.

3.3 Data Considerations

Before we examine our methodology for exploring the α-stable traffic model, we review some preliminary traffic collection concepts that facilitate understanding of the results in the following sections.

3.3.1 Data Aggregation

Throughout this work, we refer to the length of a data sample (in seconds) as the *data window*, or simply window ϖ. A window is comprised of M counting periods, or *sub-windows* Δ_{sw}. The size of ϖ and Δ_{sw} are related by

Table 3.2. Characteristics of examined datasets

Name	**Physical Network**	**Real Traffic**	**Traffic Mix**	**Labeled**	**Year**	**Average Volume** [kpps]	**Trace Length** [min]
ISCX 14 Jun	Y	N	Agent-generated, injected attacks	Y	2012	0.111	1,440
MA-CCDC	Y	N	Cyber defense scenario	Y	2010	29.8	5.6
MAWI 151114	Y	Y	Real-world backbone link	Partial	2015	156.1	15
MAWI 160428	Y	Y	Real-world backbone link	Partial	2016	109	15

$$M = \varpi/\Delta_{sw}, \tag{3.1}$$

where

$$\varpi = [t_{stop} - t_{start}], \tag{3.2}$$

and t_{start} and t_{stop} are the start and stop times of the data processing window, respectively. Note that this research uses $\varpi \in [1, 6]$ s and $\Delta_{sw} \in [1, 10]$ ms for the vast majority of quantitative as well as empirical trace analysis, with the exception of the initial model exploration and LL quantification in Table 3.3.

It is interesting that ϖ and Δ_{sw} lengths are seldom discussed in the literature. Two of the few direct comparisons are available from the previous non-Gaussian anomaly detection works discussed in Chapter 1. The α-stable detection system uses $\Delta_{sw} = 5.0$ s and $\varpi = 30$ minutes [9]. The gamma-based anomaly detection work used values for $\Delta_{sw} \in [1, 1000]$ ms and $\varpi = 1$ minute [5]. Aside from these studies, we have been unable to determine a "typical" ϖ and Δ_{sw}, likely due to the proprietary nature of such details. Based on the available documentation, our use of ϖ in the single-digit seconds appears to permit faster traffic modeling and detection than is currently described in the literature. These results are examined in further detail in Chapter 5.

3.3.2 Data Stationarity

Statistical analysis requires stationary data to generate accurate and consistent results; however, time-series data frequently cannot be proven to be either strictly or wide sense stationary, often due to additive trends, anomalies, or periodic background processes [59], [60]. Adaptations such as differencing are frequently used to de-trend data and produce a stationary result for subsequent analysis [60]. For our work, de-trending is not appropriate because the MAWI traces we have analyzed appear to possess random vice deterministic or periodic trends; this can be seen in Figure 3.5, where the changes in ZOL, a measure of average packet volume, appear to fluctuate randomly over the 480-s portion of the trace. This figure was produced by estimating the typical volumetric rate of MAWI 20160428 trace between 30–510 s using overlapping, 3.0-s windows and 4.0-ms sub-windows. The displayed volumetric rate is equivalent to average packets per sub-window, as measured by ZOL (discussed in detail in Chapter 4).

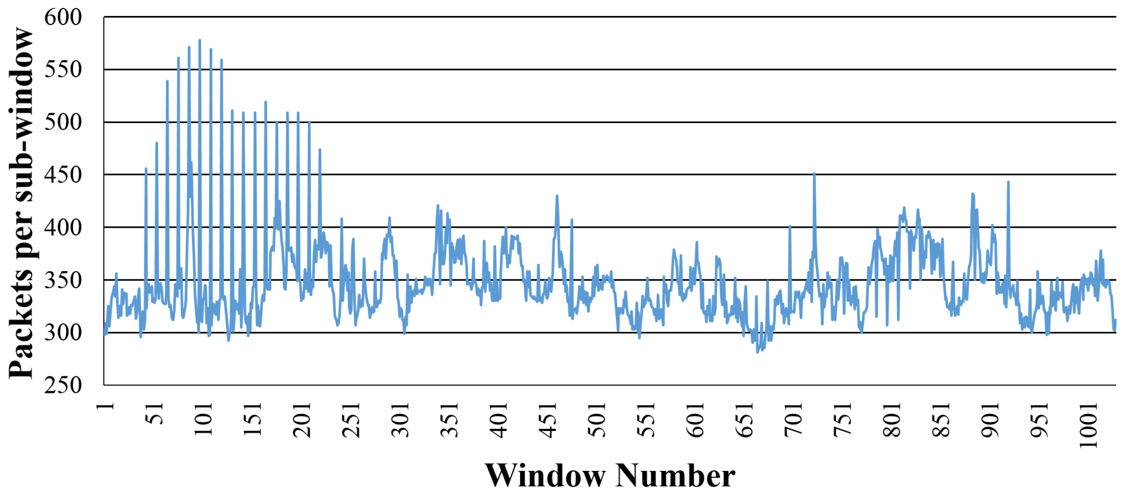


Figure 3.5. Packet volume per sub-window of MAWI trace over 480 s

Returning to the stationarity of our data samples, network traffic is broadly acknowledged to be non-stationary over very large windows, exhibiting periodicity due to human factors (e.g., the diurnal cycle) and self-similarity [2], [5], [10], [50], [61]. Similar to other approaches in the literature, our challenge becomes a problem of identifying a stationarity time limit, below which we *assume* our traffic sample is stationary.

Stationarity thresholds are not extensively discussed in network anomaly detection literature. Our stationarity assertion is supported by the sparse documentation that *does* exist, however, such that time periods of less than one hour are stationary [2], [10]. Simmross-Wattenberg et al. applied a safety factor of two to the one-hour threshold in their benchmark paper, analyzing network traffic collected in 30 minute windows. Accordingly, we begin our analysis of the MAWI traces by establishing 30 minutes as our maximum interval for assuming long-term stationarity.

This decision can be reinforced through empirical analysis of our chosen network traffic traces in terms of some of the requirements for stationarity, such as a constant mean [59]. To do so, we must first consider the short-term fluctuations of ZOL in Figure 3.5. Fluctuations such as these are well-examined in the literature and are indicative of short-term anomalies or random walks inherent to some random processes and do not contradict our assumption of stationarity [60], [62]. These random walks may become a concern during detection; however, as such, they are discussed and addressed in Chapter 5.

Discarding the random fluctuations permits assessment of the long-term trend in average packet count per sub-window. A consistent measure of the typical packet rate over the first half of the displayed trace (i.e., through window 505) appears to be 350 packets per sub-window. Assessing the second part of the trace, 350 still appears to be a reasonable estimation of this rate. Since the "average" volume fluctuates aperiodically around the same level for the length of the sample, we can hypothesize that the MAWI trace is trend-stationary for a period of at least eight minutes [62].

Accordingly, given that our window sizes (on the order of 1–6 s) are more than two orders of magnitude less than the thresholds established in the literature and the empirical assessment above, we adopt the more conservative of the two thresholds, setting an eight-minute stationarity time limit for our results obtained from the MAWI traces.

Note that while some qualitative conclusions in this chapter are reached using non-MAWI datasets sampled at window sizes in excess of 8 minutes, all quantitative conclusions regarding α-stable model and detection performance use MAWI samples with window sizes below this stationarity threshold.

Now that the traffic traces of interest as well as the methodology and assumptions for trace analysis have been discussed, it is appropriate to examine the modeling results.

3.4 Traffic Modeling Results

Given the previous work described in Section 3.1, we now explore our results and conclusions regarding the applicability of the α-stable model to network traffic. Portions of this section, including the model evaluation methodology, LL fit results, and model fit response to sub-window size, were presented at the 2018 Hawaii International Conference on System Sciences and published in the conference proceedings [20].

3.4.1 Methodology Overview

To validate the α-stable network traffic model and extend prior work [9], we examined the ISCX, MACCDC, and MAWI traces over a range of window and sub-window lengths and for the four previously-mentioned traffic scenarios. Varying window sizes produced novel results regarding the effect of data aggregation on α-stable model accuracy, while varying the traffic scenario extended prior work to demonstrate the α-stable distribution is more appropriate than alternative distributions including the Poisson, Gaussian, Weibull, gamma, and exponential for both artificial and human-generated traffic.

To reach this conclusion, we obtained representative samples from the three datasets for the traffic types specified in Table 3.3. These cases were then transformed into a vector of M packet counts per sub-window using the Python script contained in the Appendix; the sub-window and window size for each case are also shown in Table 3.3. Applying the methodology described in Section 2.3 for each case, we recorded the LL of the ML fit to the histogram of the data for each of the six candidate distributions. Note that the Poisson model performed more poorly than the exponential model and these results are not shown in the table due to space constraints. Also, for each case the "best" performing distribution (i.e., the greatest LL) is indicated by bold font.

As a general orientation to the table, Scenarios 1-9 used data from the ISCX dataset; 14 June is benign-only traffic, and 15 June traffic contains a mix of benign and attack scenarios. The MACCDC dataset was examined in scenarios 9–13; these scenarios were chosen to more fully explore the effect of window and sub-window sizing on model accuracy. Finally, we began exploring the effects of attacks on traffic histograms using the MAWI dataset in scenarios 14-17, as the MAWI data contains numerous examples of real-world attacks.

One note regarding the LL measure bears discussion: It is most appropriate to compare the

Table 3.3. Log-likelihood comparison of distributional model by aggregation period and traffic scenario. Adapted from: [20].

Trace	Date	Scenario	Sub-window	Window [min]	Log Likelihood Ratio from ML Fit					
					Stable	Weibull	Gamma	Exponential	Gaussian	Best
1	15-Jun	Benign Low Vol	60	83	**-501.8**	-537.0	-553.0	-571.3	-728.5	-501.8
2	15-Jun	Benign Mid Vol	60	83	-578.2	-575.0	**-574.3**	-583.3	-591.2	-574.3
3	15-Jun	Mid to Attack	60	61	**-950.3**	-974.6	-967.6	-2000.0	-975.0	-950.3
4	15-Jun	Attack	60	61	**-641.1**	-641.9	-650.9	-2000.0	-646.0	-641.1
5	14-Jun	Benign	10	61	-385.5	**-382.0**	-382.4	-382.5	-438.4	-382.0
6	14-Jun	Noisy	10	61	-608.8	**-600.8**	-602.8	-605.1	-719.6	-600.8
7	14-Jun	Attack	10	25	**-534.8**	-553.3	-571.9	-613.5	-803.2	-534.8
8	15-Jun	Trace 1	10	83	**-1676.2**	-1698.1	-1787.2	-1953.8	-2807.6	-1676.2
9	15-Jun	5 min window	10	5	-106.8	**-102.9**	**-102.9**	**-102.9**	-111.2	-102.9
10	MACCDC	30s Segment	1	0.5	**-175.3**	-	-	-191.3	-184.4	-175.3
11	MACCDC	30s Segment	0.1	0.5	**-699.5**	-	-	-1275.5	-1675.7	-699.5
12	MACCDC	Full Trace	1	5.6	**-2229.6**	-	-	-2398.0	-2229.6	-2229.6
13	MACCDC	Full Trace	0.1	5.6	**-10488.2**	-	-	-16239.6	-20796.9	-10488.2
14	MAWI	Benign 20150925	4	4	-10932.0	-11111.7	**-10925.8**	-13371.1	-10993.4	-10925.8
15	MAWI	Attack 20150925	4	4	**-2682.4**	-2691.5	-2688.0	-3424.6	-2683.1	-2682.4
16	MAWI	Attack 20150925	6	6	**-4036.7**	-4114.8	-4357.3	-5142.3	-4071.1	-4036.7
17	MAWI	Attack 20150925	6	6	**-2158.7**	-2247.1	-2466.0	-2783.5	-2214.9	-2158.7
Average Normalized LL (Smaller is Better)					**1.00**	**1.02**	**1.03**	**1.36**	**1.34**	

accuracy of the various models through this average, normalized metric because LL varies according to sample size. For example, it is not appropriate to compare the results from Trace 5 directly to the numerical results from any of the other traces. To enable across-scenario comparisons, we require a size-independent metric; we use the average Hellinger distance, defined in Chapter 2, for this purpose.

Before we use the Hellinger distance to optimize performance across samples, let us return to Table 3.3 and draw some additional conclusions.

3.4.2 Best Distribution Fit

Beginning with the bottom row, we see in Table 3.3 that the stable model has the best average normalized LL across the range of examined scenarios and datasets. It is also important to note that the next best distribution for our purposes is the Gaussian distribution. Even though the Weibull and gamma distributions frequently perform better than alternatives, we have decided that these distributions are not viable for our purposes. Gamma and Weibull distributions are constrained to the range of $(0, \infty)$, and zero-count instances must be excluded to apply the models: This is a data manipulation that we judge to be undesirable because zero packet counts are a reasonable outcome at small sub-window sizes or when applied to captures from lower-volume networks.

Comparing the performance of the stable model to the alternatives across the range of scenarios, we conclude that one overarching reason for the superior performance of the α-stable model is that this distribution almost always provides a reasonable fit, if not the best fit. This is likely due to its four parameters; it can shift and scale freely while also changing skew (as given by β) to reflect the presence of attack traffic, accurately locate the distribution over the mode of the data under nearly all scenarios, and capture the outliers that are frequently present in network traffic using its heavy tail (represented by α). We explore these attributes in more detail in the next section.

3.4.3 Robust Approximation through Outlier Resistance

The overall resistance of the stable distribution to outliers, as opposed to its next-best alternative from Table 3.3, are illustrated in Figure 3.6 and Figure 3.7. Both figures show how the mean (or location) of the Gaussian distribution tends to shift towards outliers. This effect is well-documented in the general statistical analysis literature. The magnitude of this shift is proportional to both the value and population of the outliers; the Gaussian is *so* sensitive that even a few benign outliers can significantly shift its location, as illustrated in Figure 3.6. The stable distribution, though, continues to accurately locate the primary mode of the data even when the magnitude of the outliers grows significantly, such as at the onset of the DoS attack in Figure 3.7.

These properties of the stable distribution, and the inherited attributes of stable-derived estimators, are harnessed using the work in Chapter 4. Let us now return to Table 3.3 and extract some preliminary conclusions that require further analysis.

3.4.4 Effects of Window Size on Model Fit

Some of the scenarios in this table were chosen to explore the effects of data aggregation on model accuracy. Our exploratory data analysis empirically showed that larger windows (and to an extent, sub-windows) produced traffic rate histograms that tended more towards a Gaussian distribution, implying an upper bound on desirable window size for α-stable modeling. At the other extreme, the scenarios of Table 3.3 in which the exponential model produces a good approximation (i.e., Scenarios 2,5, and 9) illustrate that at lower

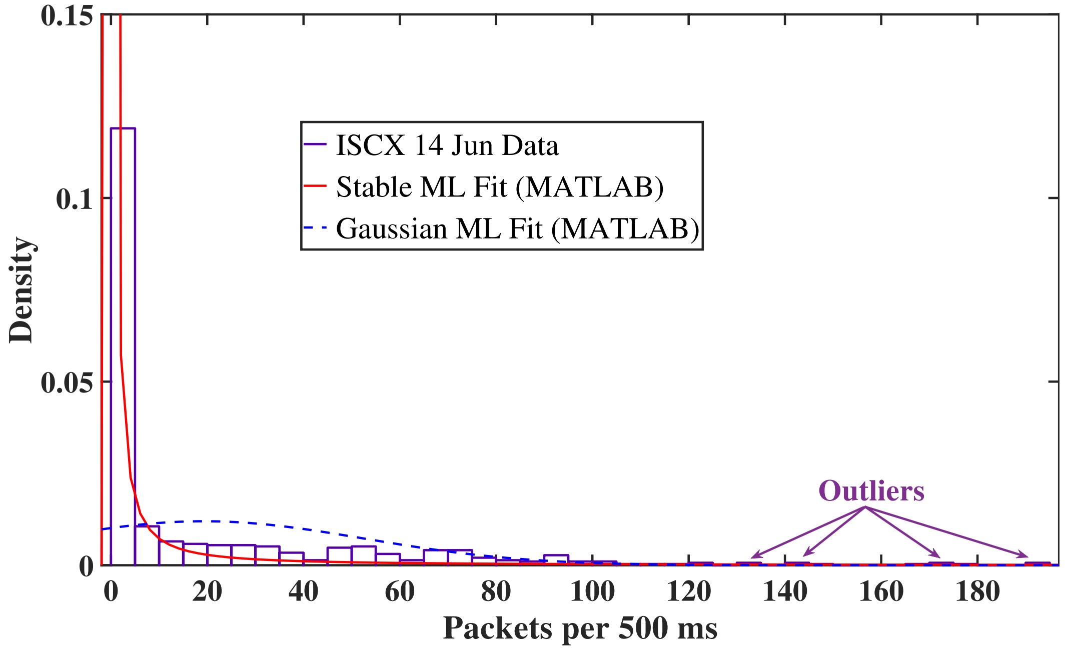


Figure 3.6. Differing effects of benign outliers on Gaussian and stable distributions, from the ISCX 14 June dataset. Source: [63].

traffic volumes or windows and sub-windows that are too small, the traffic histogram tends towards an exponential distribution due to a large number of zero (or low)-packet-count sub-windows. For instance, the best relative performance of the exponential model occurred during Scenario 9, which used the smallest window (5 minutes) and sub-window (10.0 s) of any ISCX scenario.

These empirical concepts of window boundaries and performance impacts inspired the hypothesis that the α-stable model performance could be optimized through adjusting sub-window and window size. We explored this concept through repeated analysis of different MAWI scenarios, adjusting window and sub-window size to examine how the model fit error changed. These results are shown in Figure 3.8 and Figure 3.9.

The average Hellinger distance of the stable model as a function of window size in seconds for randomly-chosen attack and benign portions of the MAWI 20151115 trace are shown in Figure 3.8. Using this distance to measure the cumulative difference between the histogram of the data and the best-fit (i.e., ML) model of the data for a given distribution, we define the smallest cumulative difference as the model with the lowest error, and thus the "best".

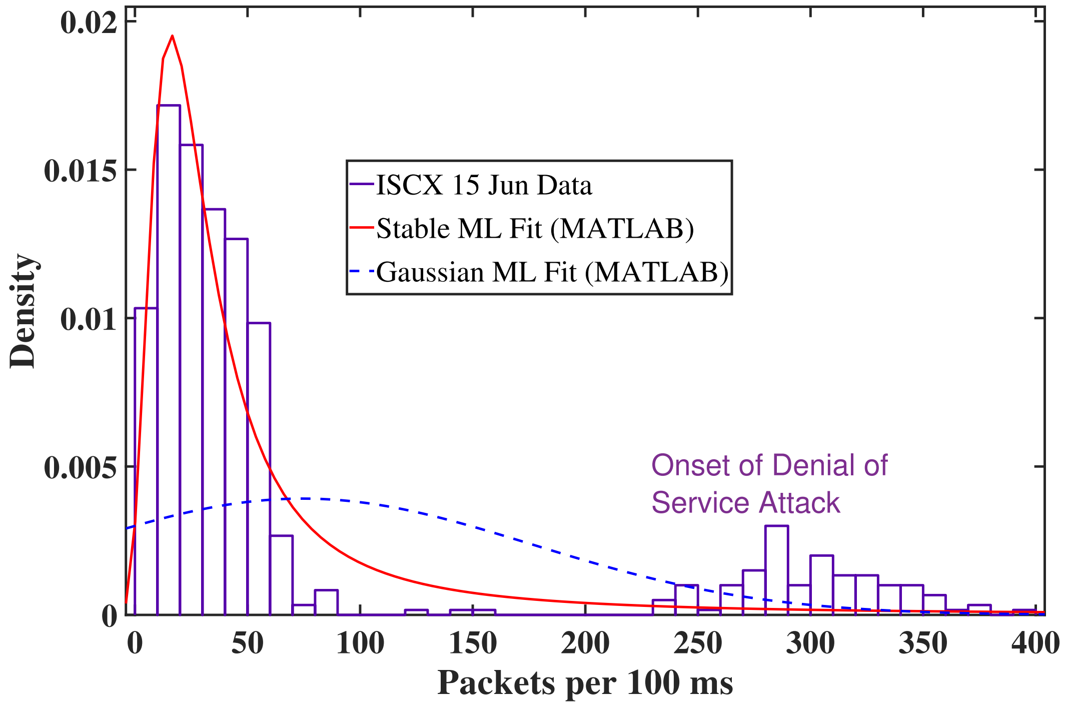


Figure 3.7. Differing effects of a DoS attack on Gaussian and stable traffic models, from the ISCX 15 June dataset. Source: [63].

One conclusion that can be drawn from this figure is that, up to a point where the window grows too small (in these cases, less than 10–15 s), fit error is relatively invariant to window size. If this trend carries across other datasets, it implies that shorter data windows can be chosen to produce the same modeling (and by extension, detection) accuracy while lowering detection system response time, which is proportional to window size. Shorter data windows are also less computationally costly in terms of collection and analysis, facilitating real-time implementation.

Turning to the effects of sub-window size on model accuracy, the average Hellinger distance as a function of sub-window size in milliseconds is shown in Figure 3.9 for a randomly-chosen attack transition portion of the MAWI 20151115 trace. Some interesting conclusions can be drawn from this figure. First, the α-stable model error has a strong global minimum; if repeated frequently across additional windows and datasets, this characteristic supports implementation of a data-adaptive network anomaly detection system. This type of adaptation has the potential to produce more accurate detection systems and has not been previously published.

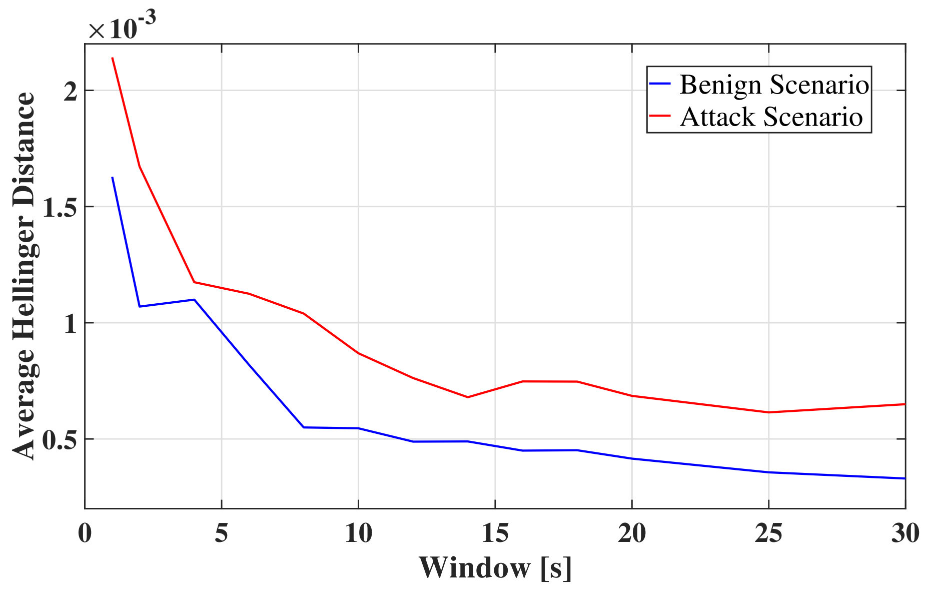


Figure 3.8. Stable traffic model error as a function of window size in seconds

Second, the Gaussian model error in Figure 3.9 seems relatively insensitive to sub-window size; if this result is also consistent, it could explain why adaptive sub-windows have not been explored in network anomaly detection. This preliminary conclusion is intuitive; per the central limit theorem, longer (vice shorter) data aggregation periods should produce traffic histograms that are more Gaussian and, thus, improved model fits.

Third, for the dataset used to create Figure 3.9, the shorter sub-windows encouraged by these results could, in turn, lead to shorter data windows that speed detection system response. A modeling or detection system built on this methodology would size data windows as a fixed product of the desired M and the optimal Δ_{sw}, such that

$$\varpi_{optimal} = M\Delta_{sw,optimal}. \tag{3.3}$$

We have completed limited exploration of the extensibility of the above conclusions for additional MAWI datasets and traffic scenario; an additional illustration of this exploration is shown in Figure 3.10. Average Hellinger distance as a function of sub-window size is again shown in this figure, this time for a randomly-chosen benign scenario of the MAWI 20151115 trace. The preliminary conclusions above are reinforced in Figure 3.10 through

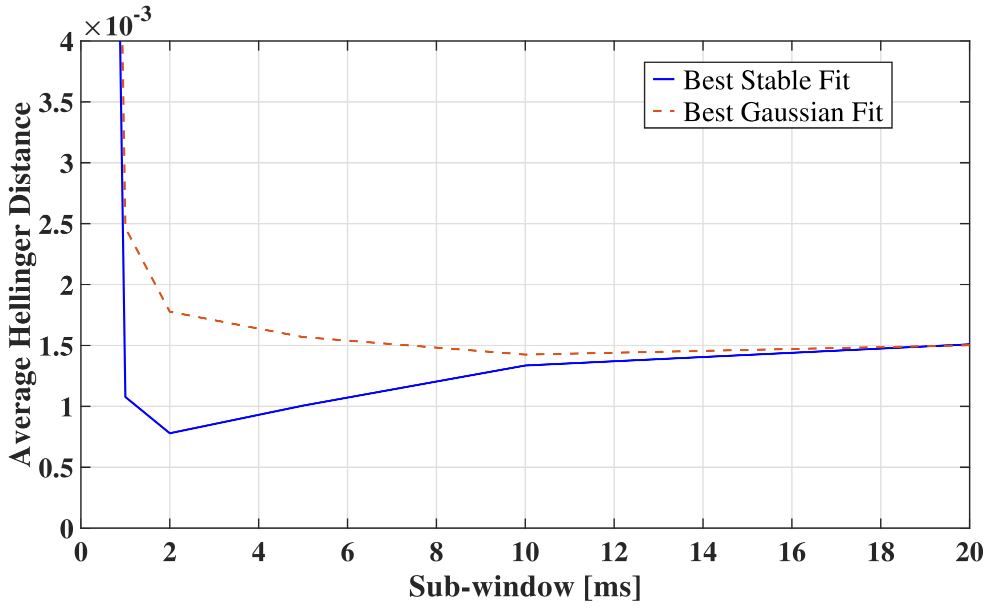


Figure 3.9. Stable traffic model error as a function of sub-window size in milliseconds. Adapted from: [20].

the presence of global optimum, a marked minimum-desirable window size, and the superior performance of the α-stable model for the examined range of sub-window sizes.

Full exploration of the applicability and gains from adaptive sub-window sizing is an item of future work. Let us continue to examine the benefits of the α-stable model through examining how the traffic scenario affects the shape of the traffic volume histogram.

3.4.5 Effects of Traffic Scenario on Sample Distribution

The investigation supporting Table 3.3 generated an additional discovery: The shape (i.e., parameters) of the stable model changes significantly during an attack beyond the shifts in location and scale that could be reasonably forecast. This exploration is important because understanding the effects of attacks on sample location and dispersion will develop intuition regarding the expected performance of the estimators that we develop and apply in our detector.

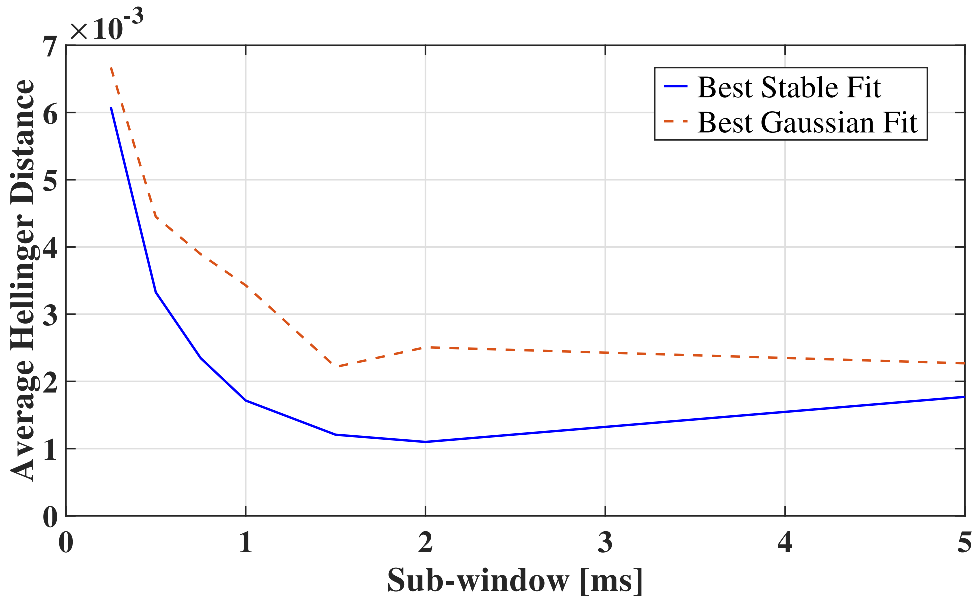


Figure 3.10. Stable traffic model error as a function of sub-window size for a benign scenario

Change in location μ during attack

The location parameter is representative of the feature count per sub-window, in our case packets per sub-window. Location μ behaves logically during an attack, with a shift that is proportional to the change in traffic rate due to the attack. The distribution of μ (i.e., the shape of the μ histogram of all the samples) also changes during an attack. We have not yet identified consistent trends across scenarios and attack types other than the volume-based rightward shift due to the additional attack traffic. We continue to analyze μ trends as we investigate our items of future work.

Change in dispersion γ during attack

The change in dispersion during an attack is more predictable than μ. Many DoS attacks and network anomalies are deterministically-generated, and the resulting traffic is relatively constant in volume when compared to the sum of the numerous random benign processes that comprise the background link traffic, as shown in Figure 3.2. Logically, the addition of this low-dispersion, high-volume anomalous traffic should reduce the overall total dispersion.

Our analysis thus far, however, shows that this result is not always guaranteed. Sample dispersion does frequently drop during an attack, following an initial *increase* during the transition period. This has been predicted through MATLAB modeling of α-stable random variable combination, as well as analysis of MAWI traffic samples. This behavior is illustrated in Figure 3.11. The per-sample dispersion of the analyzed portions of the MAWI 20151114 high-volume attack trace is plotted for each window, allowing comparison of γ values for the best STABLE-fit models of 1,472 attack and 895 benign traffic samples.

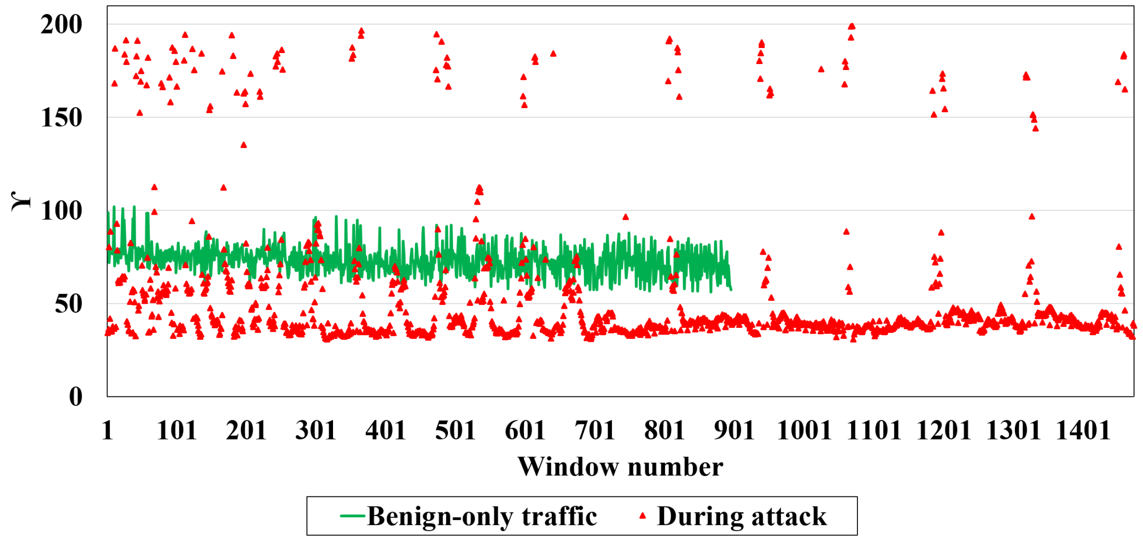


Figure 3.11. Per-sample dispersion (γ) for the MAWI 20151114 high volume attack dataset

The results in this figure clearly illustrate the change in sample dispersion that occurs during the development of an attack. For this trace, the overall dispersion of the attack traffic varies significantly because the anomaly is a deterministic ON-OFF attack, with constant amplitude during ON periods and periodic OFF periods that contain no attack traffic. (The characteristics of this attack are explored in more detail in Section 5.5.5.) These ON-OFF periods generate many transition windows, where the sample dispersion increases significantly as the sample contains varying percentages of both low-volume background traffic and high-volume attack traffic. The transition windows can be seen in Figure 3.11 to be the samples with high gamma values (e.g., $\gamma > 100$). Once the collection window "slides" to contain only attack traffic, sample dispersion drops below that of benign traffic due to the dominating contribution of the deterministic attack (e.g., the samples with $\gamma < 50$).

Note that both the sign and magnitude of the change in γ due to an anomaly depends on its type and volume; in the low-volume attack case of the MAWI 20160428 trace, γ *increases* during the attack, as shown in Figure 3.12. This appears to be because the lower-volume, varying-rate attack only spreads the mode of the traffic histogram instead of shifting it (e.g., the high volume attack case), leading to an overall increase in γ. The most interesting observation for the stable attribute estimators (SAEs) we develop in Chapter 4, as well as future work, is that the vast majority of the attack samples have γ values significantly different from the benign samples, even though the attack in this scenario was at a much lower rate (approximately 20% of the benign traffic rate).

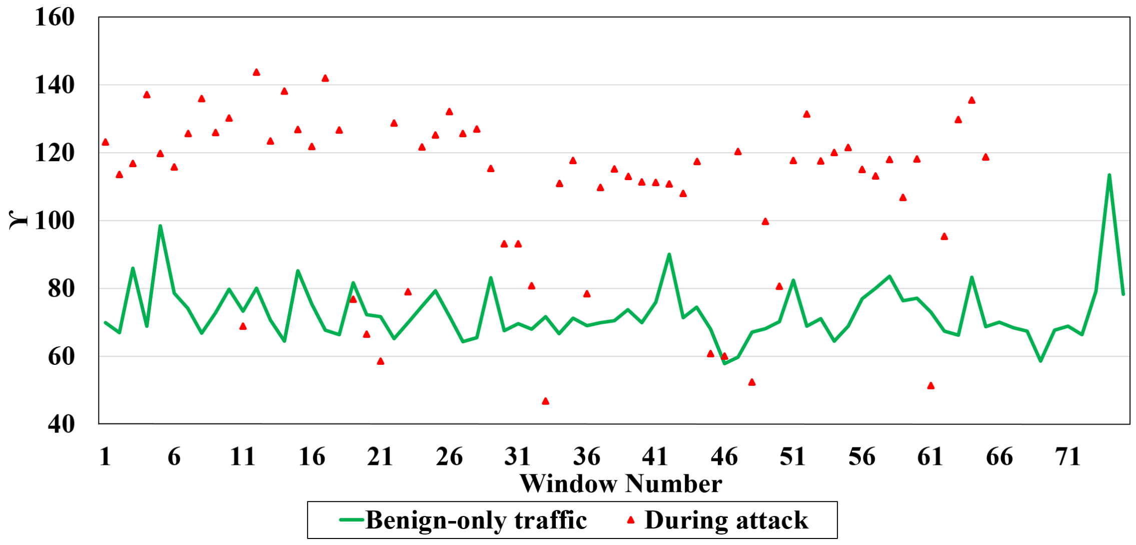


Figure 3.12. Per-sample γ in the case of a low-volume attack

Further analysis of γ effects for attacks of various types and volumes is required. This analysis will allow us to draw more definitive conclusions regarding the expected change in γ; however, the observed changes in γ during attacks and its marked difference from benign values in the vast majority of samples imply significant potential for an α-stable parametric anomaly detection system based on measuring the change in the traffic shape (i.e., parameters). The detection system that was developed for this work was designed to be scalable, so we avoided the computational costs of fitting an α-stable distribution to each sample. An accuracy and performance comparison to that of our implemented system is also promising future work.

The most critical result of the analysis in this sub-section is that γ values during an attack are, in the vast majority of cases, significantly different than those of benign samples. For our non-parametric detector, this implies that an estimator based on the dispersion of the sample could be highly effective at discriminating between attack and benign cases. Let us continue developing our intuition by examining corresponding changes in tail size α and skewness β caused by attack traffic.

Changes in α and β

Attack effects on the sample β are inconsistent. One frequent outcome is a side-effect of the previously-discussed shift in μ. During any subsequent samples where the attack traffic rate drops (e.g., an OFF period for the MAWI 2015114 attack or, for a constant attack, a lull in traffic due to probabilistic network or host effects) the background, lower-volume benign traffic appears as a heavy *lower* tail in the traffic histogram. This effect can be seen in Figure 3.13 and is indicated by negative values of β.

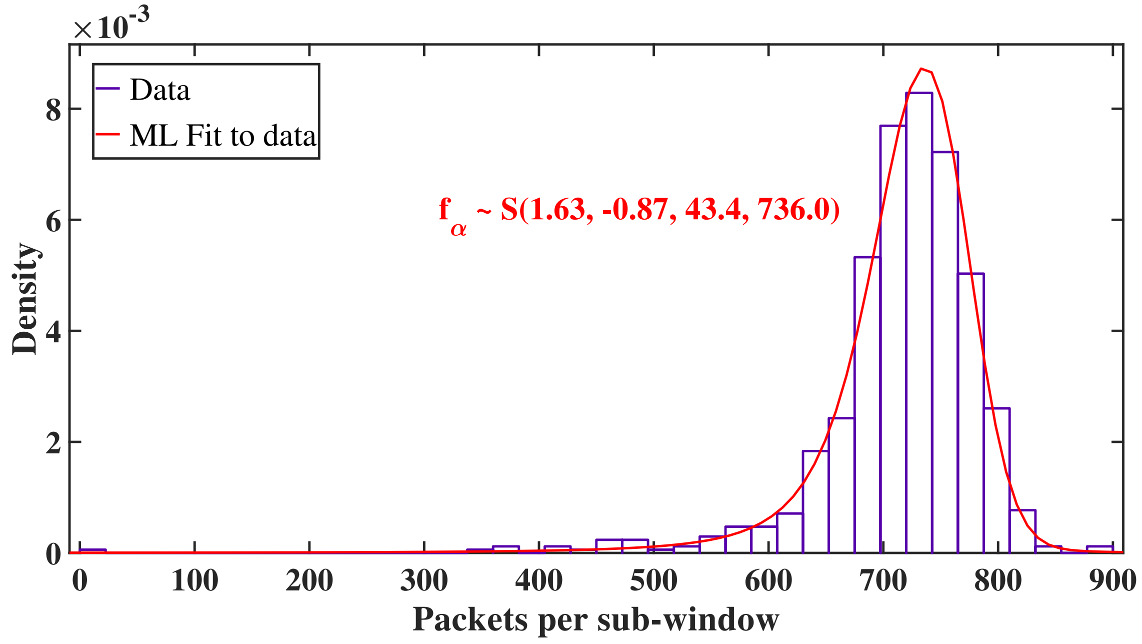


Figure 3.13. Shift in skew of the traffic rate histogram during an attack transition

This figure is from an attack transition period just following an OFF cycle and contains only a small portion of benign background sub-windows, which appear in Figure 3.13 as the outliers on the left side. Continuing to read the histogram to the right, we see the short-ramp-up period of the attack to the left of the histogram mode, with the vast majority of sub-windows concentrated around the ON-portion average rate.

Trends in β for attack and benign traffic over 14 minutes of the MAWI 20160428 trace are shown in Figure 3.14. This case was analyzed using settings of $\varpi = 6.0$ s and $\Delta_{sw} = 5.0$ ms, which produced 140 samples.

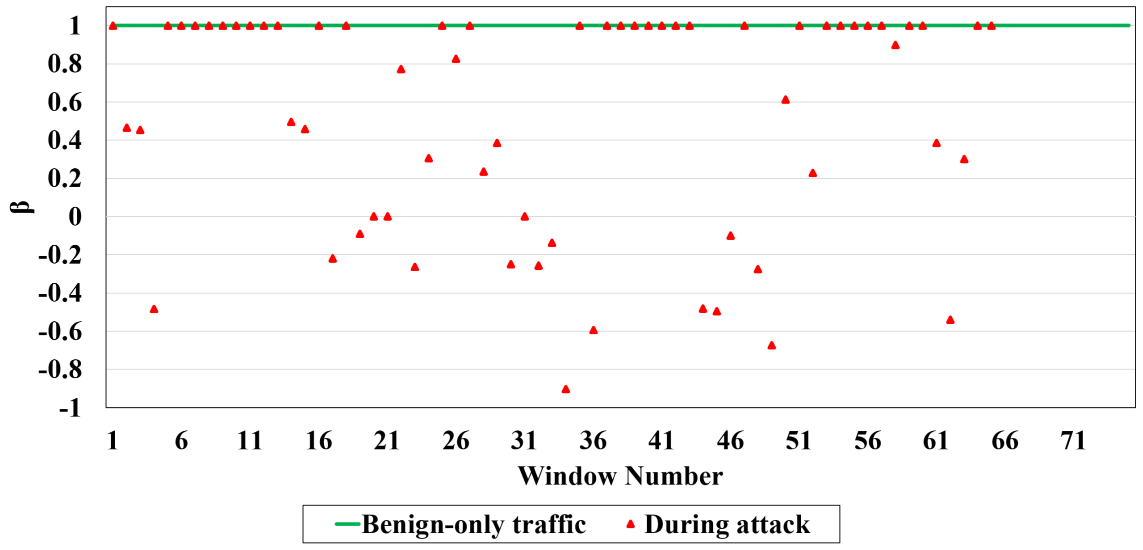


Figure 3.14. Trend in β for 14 minutes of the MAWI 20160428 trace

The best discriminator between attack and benign traffic may be that of $\beta \neq 1$, because nearly all benign samples in both the high and low volume attack scenarios have β values of 1. During an attack, β becomes a less valuable indicator than α because the attack traffic histogram tends to become Gaussian or near-Gaussian. For our purposes, we define a traffic sample with $\alpha \geq 1.99$ as Gaussian.

As this happens, the value of β becomes unimportant (theoretically, β is undefined for $\alpha = 2$ [14]). From an analytic standpoint, the ML estimation algorithms of both MATLAB and STABLE also become less reliable around our empirically-determined threshold of $\alpha = 1.99$; above this point, the β values can fluctuate significantly between repeated estimates, though the α, μ, and γ estimates are largely convergent. This inconsistent estimation of β in near-Gaussian cases is one issue that must be addressed in a parametric-detection implementation but is not important for the purposes of this work. If anything, this encourages a parametric implementation based on a combination of β and α, so let us now examine the low-volume attack effect on α, as shown in Figure 3.15.

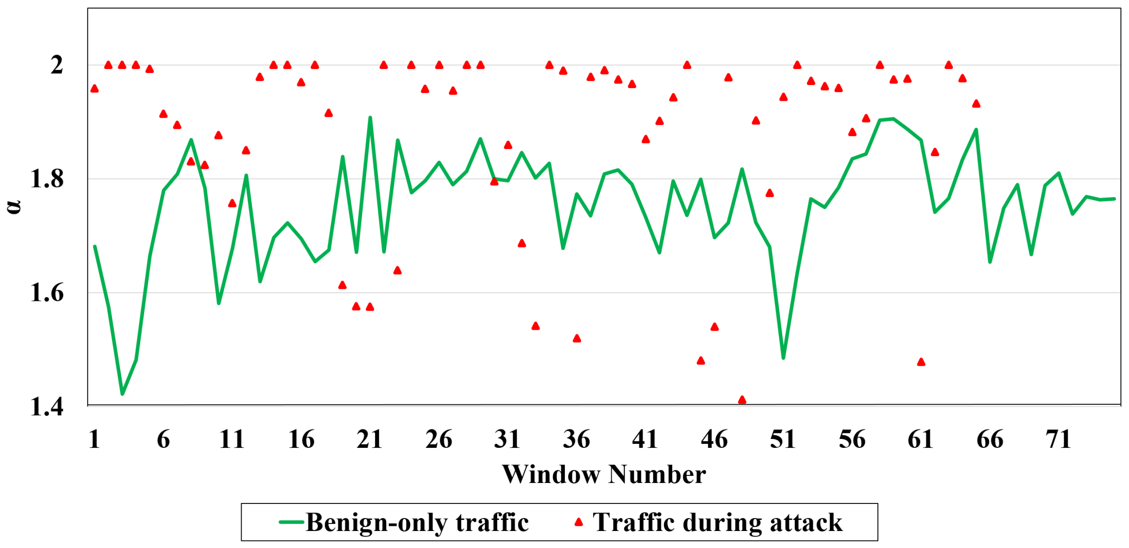


Figure 3.15. Trend in α for 14 minutes of the MAWI 20160428 trace

This figure was also developed using 14 minutes of the MAWI 20160428 trace with analysis settings of ϖ = 6.0 s, Δ_{sw} = 5.0 ms, and Δ_o = 0. Using the empiric α threshold above, we can see that the attack traffic is frequently Gaussian but more often non-Gaussian. We can also see that the benign traffic is *never* Gaussian.

Combining the attack and benign sample results confirms that the *vast* majority of traffic in the MAWI20160428 trace is α-stable ; specifically, 86.4% of the samples have a STABLE ML fit where $\alpha < 1.99$ (121 of 140). To validate this result, we analyzed the trend in α for the analyzed portions of the MAWI 20151114 trace using sample settings of ϖ = 3.0 s, Δ_{sw} = 4.0 ms. 892 of 895 (99.7%) benign samples and 1357 of 1,472 (92.2%) attack samples had values of $\alpha \leq 1.99$ (i.e., were α-stable vice Gaussian). Combining these two results, we see that network traffic is dominantly and repeatedly α-stable -distributed. This has significant implications for model appropriateness and accuracy in the general field of network traffic monitoring, particularly for proprietary networks that are not frequently exposed to attacks or anomalies. This result also validates our intended extension of α-stable methods to the anomaly detector, which is the subject of the next chapters.

3.5 Summary

In this chapter, using three public datasets analyzed for a range of aggregation periods and traffic scenarios, we have shown that the best model for the marginal distribution of network traffic packet rate is the α-stable distribution. For our empirical threshold of $\alpha \leq 1.99$, benign network traffic was shown to be nearly entirely α-stable, while traffic during an attack is α-stable in excess of 80% of the time. Given that for many networks, attacks are the great exception rather than the rule, this result proves that the α-stable model should be given strong consideration as the **standard** for network traffic.

Additionally, we showed that the α-stable parameters of α, β, and γ change significantly when attack traffic is added, even at relatively low volumes of 15–25% of background traffic. (It was not shown, but it *is* intuitive that μ changes as well, proportional to the volume of traffic added.) Together, these changes in parameters imply potential for an accurate anomaly detection system based on observed changes in the ML α-stable fit.

Finally, we demonstrated that the fit of the α-stable model to the traffic data is dependent on sub-window size and relatively independent of window size. This implies that there should be little accuracy degradation using small data windows to speed detection system response and that it should be possible to build an α-stable anomaly detector that optimizes sub-window size to improve accuracy.

The challenge, and the objective of the remainder of this work, becomes using these findings regarding α-stable traffic modeling to design a more accurate network anomaly detection system. We must first examine, in detail, the implications of traffic following an α-stable distribution. Keeping these implications in mind, we can use the literature to develop α-stable approximation or estimation methods that enable a real-time detection system. Then, after building an end-to-end detection system, we can finally explore and compare the performance of a non-Gaussian approach to network anomaly detection.

CHAPTER 4: Estimation and Approximation of α-Stable Data

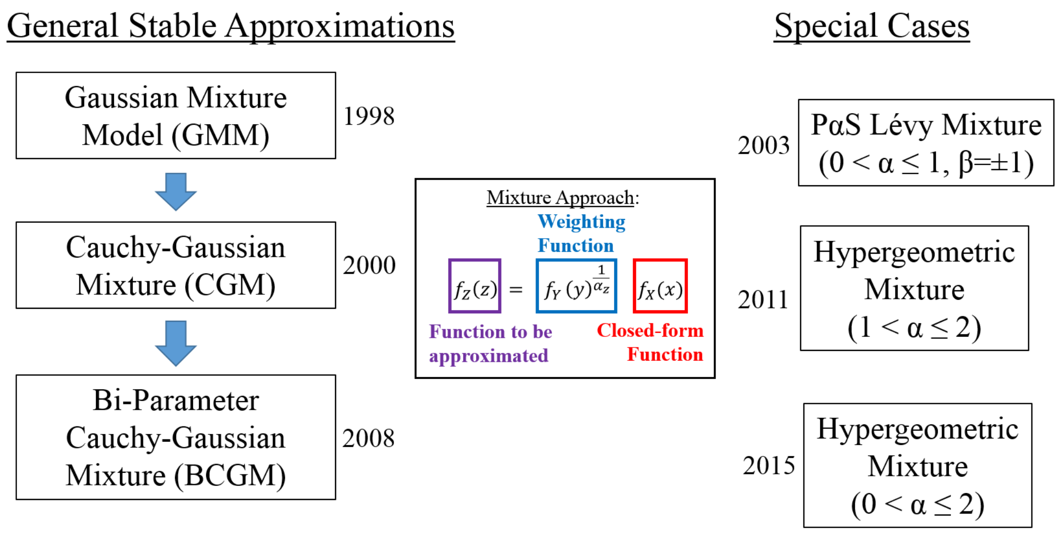
In this chapter we examine various approaches to formulating the α-stable signal models that are used in our detection system. Many traditional detection approaches use probability or likelihood ratios based on the PDFs of the data samples. To obtain this PDF, we can use closed-form solutions (should they exist), approximations, or estimates of key attributes of the best-fit distribution. We examine closed-form solutions and approximations and find that they either cannot model the full range of our data or their limitations on our intended detection system exceed any accuracy advantages they provide.

Alternatively, estimates of the PDF derived from the data instead of an MLE-fitted distribution provide an intriguing option. These *stable estimators* determine representative, stable-parameter-like *attributes* from the data sample. Stable estimators are shown to be a lightweight (i.e., computationally-efficient) and accurate alternative for implementation in our detection system. We begin our discussion with a review of the potential and limitations of the stable solutions and approximations most suited to our problem, then proceed to the stable estimators.

4.1 Stable Signal Modeling Approaches

Numerous methods have been developed that seek to provide closed form solutions (or approximations to these solutions) in order to harness the accuracy and flexibility of the α-stable distribution while avoiding the computational complexity brought about by numerical estimation methods.

The evolution of these approaches is shown in Figure 4.1; all methods in this figure use weighted mixtures of closed-form distributions to approximate an α-stable function after its parameters $\boldsymbol{\theta}$ have been estimated from the data. This methodology leverages the closed-form expressions of their components while still providing close fits to the data [29]. Note that the special case approximations given on the right side of Figure 4.1 are constrained from the maximum permissible range of $\boldsymbol{\theta}$ as shown.



General Stable Approximations
Special Cases
Gaussian Mixture Model (GMM)
1998
Cauchy-Gaussian Mixture (CGM)
2000
Bi-Parameter Cauchy-Gaussian Mixture (BCGM)
2008
Mixture Approach:
Weighting Function
$f_Z(z) = f_Y(y)^{\frac{1}{\alpha_z}} f_X(x)$
Function to be approximated
Closed-form Function
2003
PαS Lévy Mixture $(0 < \alpha \leq 1, \beta = \pm 1)$
2011
Hypergeometric Mixture $(1 < \alpha \leq 2)$
2015
Hypergeometric Mixture $(0 < \alpha \leq 2)$

Figure 4.1. Evolution of α-Stable approximation methods

The special case approaches were of interest because they have the potential to enable closed-form solutions using non-Gaussian components which, in turn, may enable non-Gaussian detection algorithms. The hypergeometric approaches promise closed-form solutions that are computationally-tractable and accurate. In the next sections, we discuss these non-Gaussian approaches to α-stable representation and approximation.

4.2 Hypergeometric Approaches

The use of hypergeometric functions to develop closed form, finite sum solutions of α-stable distributions is an area of active work in the literature. Multiple α-stable hypergeometric approximations have been developed since the 1990s, and considered earlier, for particular values of β and α [64]. The most general hypergeometric approach was published in 2015.

4.2.1 Generalized Hypergeometric Solution for $\alpha \in (1, 2]$

A generalized closed-form solution for α-stable distributions was developed in 2015 [65] but was restricted to values of $\alpha \in (1, 2]$. While this is most frequently the range of MAWI network traffic at our chosen windows and sub-windows, other traces we examined were best fit using models with $\alpha \in (0, 1)$ (e.g., ISCX).

Additionally, this approach constrains β such that

$$\left|\beta \tan\left(\frac{\pi\alpha}{2}\right)\right| < 1 - \alpha^{-1} \tag{4.1}$$

which results in limiting assumable β for some α values of interest, per Table 4.1.

Table 4.1. Constraints on values of β given α using the closed-form hypergeometric solution approach of [65]

min(β)	α	max(β)
0.0	1.1	0.0
-0.1	1.2	0.1
-0.1	1.3	0.1
-0.2	1.4	0.2
-0.3	1.5	0.3
-0.5	1.6	0.5
-0.8	1.7	0.8
-1.4	1.8	1.4
-3.0	1.9	3.0

The vast majority of our benign MAWI data is best-fit by α-stable distributions with $\alpha \in [1.4, 1.9], \beta = 1$; thus, per the limits in Table 4.1, this hypergeometric approach cannot model skewed network traffic data unless $\alpha > 1.7$. This restriction eliminates large portions of our benign data *and* would significantly restrict the flexibility of our detection system; for these reasons it is necessary to consider other hypergeometric approaches.

4.2.2 Generalized Hypergeometric Solution for $\alpha \in (0, 2]$

In 2011, a hypergeometric closed-form finite sum solution was published that models the entire permissible range of α. This solution decomposed α-stable RV Z into weighted finite sums of gamma distributions per

$$f(\alpha, \beta; z) = \sum_{j=1}^{M-1} \frac{c_j(l, k, r)}{x^{1 \mp jl/M}} \, {}_{m+1}F_M \left({}^{1, \Delta(m, 1+jm/M)}_{\Delta(M, j+1)} \middle| (-1)^{r-M} \frac{m^m x^{\pm l}}{M^M} \right) \tag{4.2}$$

where

$$c_j(l,k,r) = \frac{M^{1/2-j} m^{1/2+jm/M}}{2^{-r}(2\pi)^{(l+k)/2}} \left[\prod_{i=0}^{m-1} \Gamma\left(\frac{j}{M} + \frac{i+1}{m}\right)\right] \frac{\left[\prod_{i=1}^{j} \Gamma\left(\frac{i-j-1}{M}\right)\right]\left[\prod_{i=j+2}^{M} \Gamma\left(\frac{i-j-1}{M}\right)\right]}{\prod_{i=0}^{r-1}\left[\sin\left(\pi\frac{i}{r} - \pi\frac{j}{M}\right)\right]^{-1}}, \tag{4.3}$$

for integers l and k such that $l < k, \alpha = l/k$; for integer $r \in (0,l)$, $\beta = \alpha - 2r/k$; and $min(l,k) = m < M = max(l,k)$ [64].

There are also restrictions to this approach. The broader hypergeometric solution of (4.2) is only applicable for integer l and k that produce a rational α. Additionally, the value of β must be defined using r,k, and α. Similar to the hypergeometric $\alpha \in (1,2]$ approach, these constraints somewhat reduce the solution's potential flexibility (and, thus, accuracy) to represent our fitted data.

Another equally-serious issue exists: the computational cost of hypergeometric approaches. As these approaches use potentially large sums of gamma distributions, the solution has the potential to become costly, particularly if large l, k are used to develop a close approximation to α. The high cost of hypergeometric approximation is illustrated in Figure 4.2, where for varying values of α, the execution time of both discussed hypergeometric approaches as well as a reference numerical approximation method are displayed [65]. In this figure, the costs of the reference method are annotated in black, while the costs of Pogany and Nadarajah's approach [65] is annotated in red and Gorska and Penson's [64] in blue. Note that while both hypergeometric approaches are faster than a commonly-used numerical approximation, they provide only a $10 - 20\%$ improvement over the typical MLE approach.

In sum, hypergeometric solutions incur significant costs through using a gamma distribution as well as the costs of first estimating the α-stable parameters $\boldsymbol{\theta}$. Additionally, their constraints restrict the applicability of the method to a space smaller than the permissible range of α-stable parameters. For those reasons, we consider it appropriate to consider other approximation and estimation methods and now turn to the PaS approximation method. Much of this next section was presented at the 2017 Asilomar Conference on Signals, Systems, and Computers and is scheduled for publication in its proceedings [63].

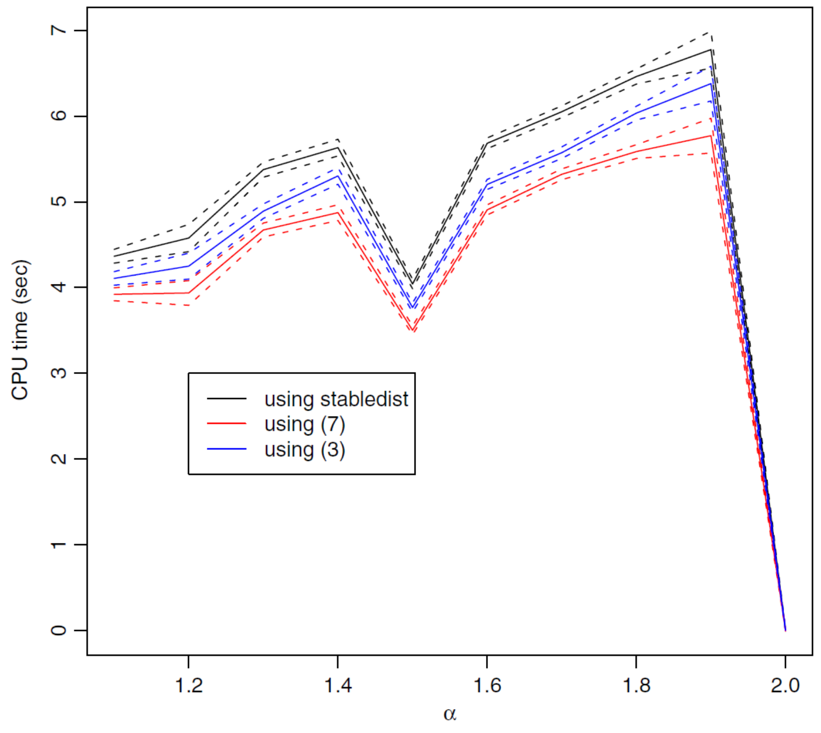


Figure 4.2. Comparison of computational time required by three methods for approximating an α-stable PDF. Source: [65].

4.3 Positive α-Stable

Instead of a weighted mixture of Gaussians, the PaS methodology approximates a suitable α-stable RV (constrained per Figure 4.1) using a weighted mixture of Lévy RVs distributions [30]. This LMA approach is advantageous because in the Lévy special case of $\alpha = 0.5$, the PDF can be defined in closed form as discussed in Section 2.2.3.

There was no work in the literature subsequent to Kuruoglu's promulgation of the decomposition theory [30], which also did not include detail regarding implementation. Our research into the feasibility of using this method in our detection system yielded a novel algorithm for PaS approximation and novel findings regarding the numbers of methodology of determining the mixture components for data samples with distributions limited to the range of $\alpha < 0.5$

4.3.1 PaS Decomposition

The theoretical relationships for decomposing a PaS RV into the product of another PaS RV and a Lévy RV were developed by Kuruoglu based on work by Hardin [30], [66].

The decomposition of Z for $\alpha_Z \in (0, 1)$ can be expressed as

$$Z = XY^{1/\alpha_X} \tag{4.4}$$

where Y is α-stable with $\alpha_Y \in (0, 2]$ (the *mixing function*) and X is distributed per (2.4) [30].

The PDF of Z can then be discretely approximated by

$$f_Z(z) \approx \frac{1}{C} \sum_{i=1}^{N} \left(\frac{f_Y(y_i)}{y_i^3} f_X\left(\frac{z}{{y_i}^2} \right) \right) \tag{4.5}$$

where

$$\gamma_Y = \frac{\gamma_Z}{2^{\alpha_Z} \gamma_X^{2\alpha_Z}} \frac{\cos(\pi \alpha_Z)}{\cos(\pi \alpha_Z/2)}, \tag{4.6}$$

$$\alpha_Y = \alpha_Z/2, \tag{4.7}$$

C is a constant, i is an integer $\in [1, N]$, and N is the number of sampling points (as well as mixture components) [30].

Examining (4.5), we find that the left terms inside the summation are weighting constants determined by choosing sample points y_i, while the right terms are Lévy RVs [30]. Both terms are scaled using the sample points and relationships shown in Figure 4.3.

The normalization constant C is given by

$$C = \sum_{i=1}^{N} \frac{f_Y(y_i)}{2y_i}. \tag{4.8}$$

Note that (4.5) applies to the case where $\alpha_Z \in (0, 0.5)$; similar expressions exist for $\alpha_Z \in (0.5, 1)$ [30]. In this section, we only consider the case of $\alpha_Z \in (0, 0.5)$ for reasons discussed later.

4.3.2 LMA Algorithm

To develop our LMA algorithm and formula for C, we extended the relationships and theory in [30] using similar approaches in the literature, such as for mixtures of Gaussians [29].

As an input, the LMA algorithm requires the parameters of a PaS RV Z with $\alpha_z < 0.5$. From Z, the mixing function $f_Y(y)$ is then generated, which is used to obtain N weights and scaling factors y_i and $f_Y(y_i)$.

These weights and scaling factors are in turn applied to N Lévy functions; the weighted sum of these *components* approximates the original distribution of Z. The LMA process for $\alpha \in (0, 0.5)$ is summarized in Algorithm 1.

Algorithm 1 LMA Algorithm for $\alpha \in (0, 0.5)$. Source: [63].

```
Inputs: Data and N
Output: LMA of $f_Z(z)$
 1: Fit $f_Z(z)$ to data, confirm $\alpha_Z < 0.5$
 2: Compute $\alpha_Y$,$\gamma_Y$ from $\alpha_Z$,$\gamma_Z$ using (4.6),(4.7),$\gamma_X = 1$
 3: Generate mixing PDF $f_Y(y)$
 4: Assign sample points $y_i$ for $i \in [1, N]$
 5: for $y_i$ do
 6:     Compute $f_Y(y_i)$
 7:     Generate scaled Lévy component $f_X(z/y_i^2)$
 8: end for
 9: Compute $C = \sum_N \left(f_Y(y_i)/(2y_i)\right)$
10: return $\hat{f}_Z(z)$ using (4.5)
```

An illustration to assist in visualizing the purpose of the mixing function is contained in Figure 4.3. As shown in this figure, the LMA weight $f_Y(y_6)$ is obtained from the appropriate mixing function for the hypothetical case of $f_Z(z) \sim S(0.4, 1, 1, 0)$.

The finite sum of Lévys resulting from the process in Algorithm 1 is the LMA. To determine the relative accuracy of this approximation, we measured average error between the data histogram and LMA using the Hellinger distance as given in Section 2.4. The built-in MATLAB ML fitting and histogram functions were used for estimations and results; the interested reader is referred to the source documentation for additional implementation detail [39].

Our preliminary investigation into LMA identified two novel conclusions regarding the

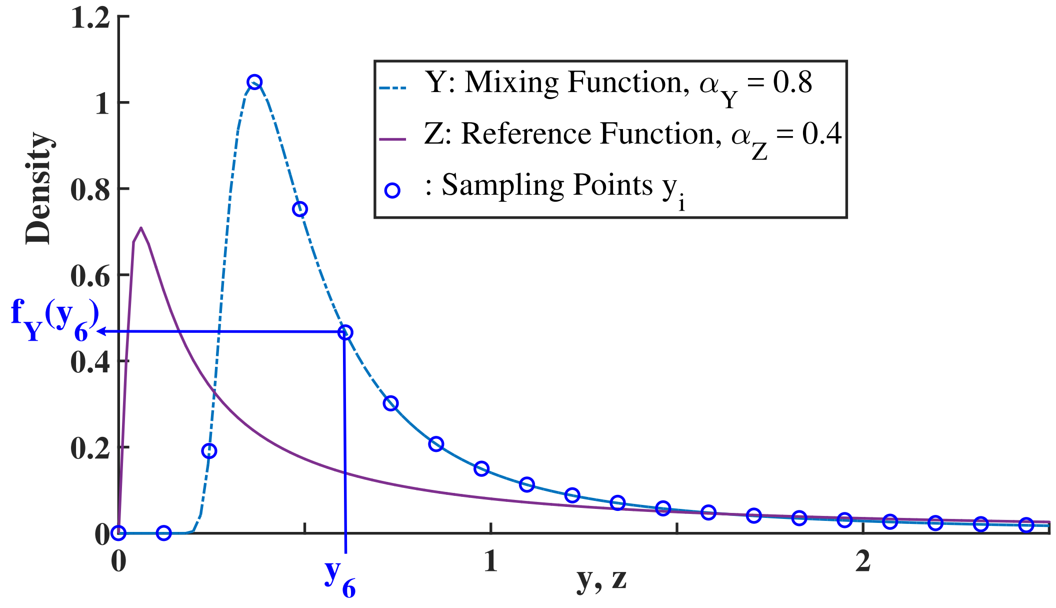


Figure 4.3. Method of sampling the mixing distribution to determine weighting constants $f_Y(y_i)$. Source: [63].

effects of sample point placement and the number of sample points required to obtain an accurate approximation.

4.3.3 LMA Investigation: Importance of sample point location

Gaussian mixture approaches recommend uniform sampling of the mixing RV Y at a large number of points N or using a post-processing algorithm to optimize the sample point placement [29]. The guidance for the PaS approach is similar [30].

Our results demonstrate that, at least for the PaS method, approximation error is relatively independent of N and largely dominated by sample point placement. It, thus, becomes possible to improve computational efficiency and overall accuracy by selectively applying sample points using either linear sampling around the peak of the reference distribution or via implementation of a lightweight sample point placement algorithm.

These results are illustrated in Figure 4.4, where the density functions for LMAs with two different values of N are compared. Case (b) was obtained using only $N = 6$ components and is more accurate than the 60-component Case (a) approximation in terms of average

Hellinger distance; visually, the fits of both results are essentially indistinguishable at the selected scale.

In Figure 4.4, we see that sample point placement is more important than the number of points in determining LMA accuracy. Case (a) samples were uniformly distributed $\in [0, 1]$, while Case (b) samples were uniformly distributed $\in [0, 1.2]$: The average d_{Hel} were 5.09×10^{-2} and 5.00×10^{-2}, respectively. The slight reduction in fit error of Case (b), while using fewer points, was due to shifting sampling bounds and the resultant change in sample point location.

4.3.4 LMA Investigation: Ability to optimize sample location

The previous result encouraged exploration of optimal sample point placement routines. We found that when using small N, non-uniform sampling of the mixing function can be utilized to improve approximation accuracy, as shown in Figure 4.5.

Based on qualitative assessments as well as Figure 4.5, the best approximation accuracy is generally obtained by concentrating sampling points around the peak and areas of maximum change of $f_Y(y)$ (i.e., points y_3, y_4, and y_5 in Figure 4.3). Other methods were examined such as sampling only the left or right tails; these alternatives produced inferior results and were omitted from Figure 4.5 for display purposes.

The results in Figure 4.5 reinforce the observation from Figure 4.4 that accuracy is relatively invariant to the number of components (at least as long as the sample points are located appropriately). The significant variation of the linear method fit error with $N \in [5, 10]$ is due to the changing sample locations, as seen previously.

Overall, the results in Figure 4.4 and Figure 4.5 show that post-processing is not strictly necessarily to fine-tune the sample locations, though this added step could be used to improve the relative accuracy for a given error measure, as done for the LMA in Figure 4.6.

The end-to-end approximation result of applying the LMA algorithm to the data in the ISCX trace modeled in Chapter 3 is shown in this figure. The sample was drawn from the 14 Jun dataset. In this case only four sample points were used, though their placement was manually optimized. The resulting $d_{Hel} = 4.32 \times 10^{-2}$, smaller than the 6 *and* 60-component LMAs cases.

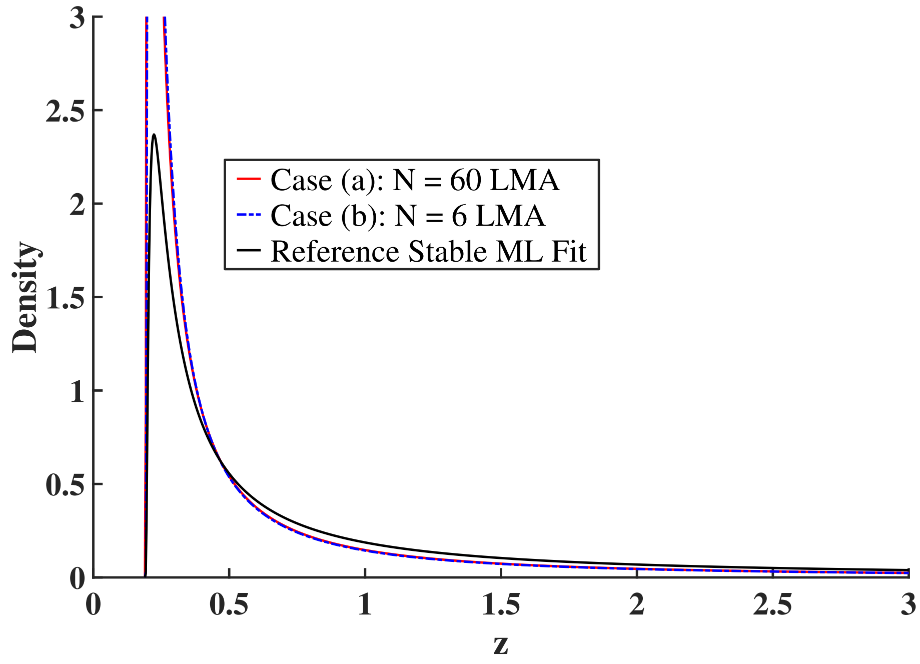


Figure 4.4. Demonstration of the insensitivity of LMA accuracy to the number of components N. Source: [63].

Additional investigation of sampling optimization methods is an item of future work and should allow quantitative evaluation of cost-accuracy trade-offs. Based on qualitative evaluation, accuracy is optimized by alternating sample points around the mode of the data (or distribution). Concentrating sample points on the upper tail side of the mode will improve the upper tail fit of the approximation while worsening the lower tail fit and, frequently, the accuracy around the peak. The reverse applies for concentrating samples on the lower tail side. Finally, concentrating samples around the peak can maintain approximation accuracy when using a smaller N but appears to lead to overestimating the mode (as can be seen in Figure 4.6).

4.3.5 LMA for $\alpha \in (0.5, 1)$

Overall, the PaS approximation theory is applicable to distributions with $\alpha \in (0, 1)$ [30]. As previously identified, the LMA algorithm and relationships in this section apply only to the case of $\alpha < 0.5$. For $\alpha > 0.5$, the theory relies on different transformations and, thus, requires a different LMA algorithm.

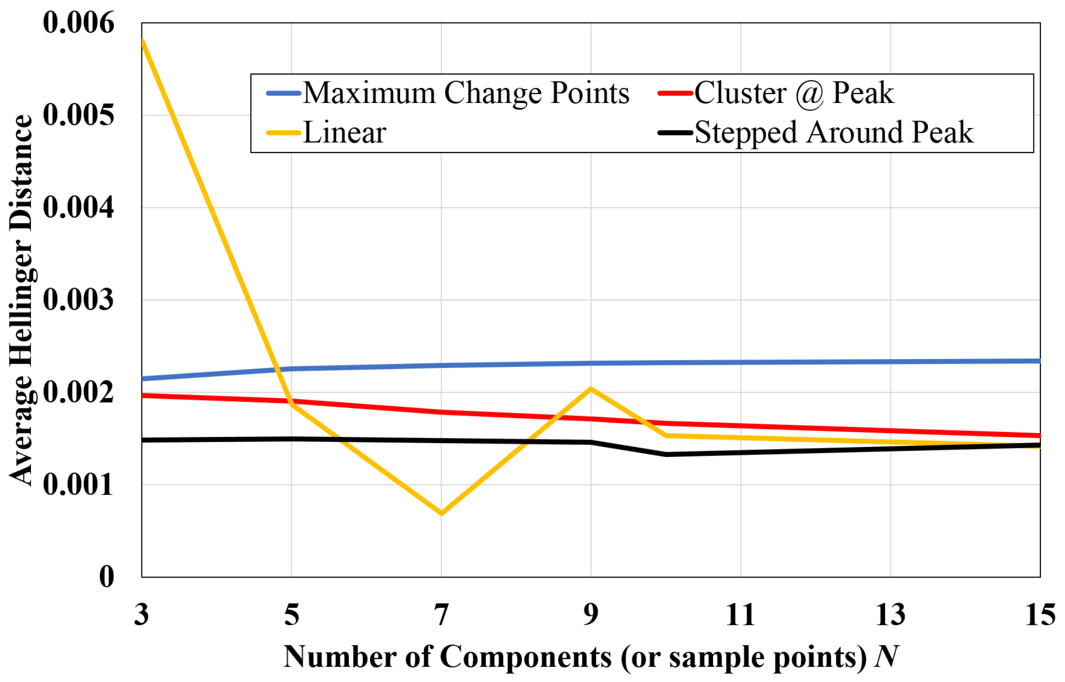


Figure 4.5. Comparison of fit errors with varying sample point location schemes and N. Source: [63].

After developing the methodology for $\alpha < 0.5$ and beginning to apply it to the selected datasets, the estimated α values revealed that LMA would not be capable of modeling the dataset of interest (i.e., the MAWI traces). The vast majority of analyzed MAWI samples were characterized by α values $\in [1.2, 2]$, as illustrated for the MAWI 20160428 trace in Figure 3.15. Accordingly, exploration of LMA was suspended in favor of focusing on efforts that were suitable for analyzing MAWI data and that would enable the detection system to operate over the full range of α observed in the real world. Further exploration of the LMA method over the full range of α will be an item of future work.

4.3.6 Summary of Hypergeometric and Approximation Approaches

We have examined the approaches in Figure 4.1 and found that they are not suited for our chosen application due to a combination of modeling constraints and computational costs. Fundamentally, these methods *all* incur:

- Cost to initially estimate an α-stable fit to the data
- Cost to determine the approximation

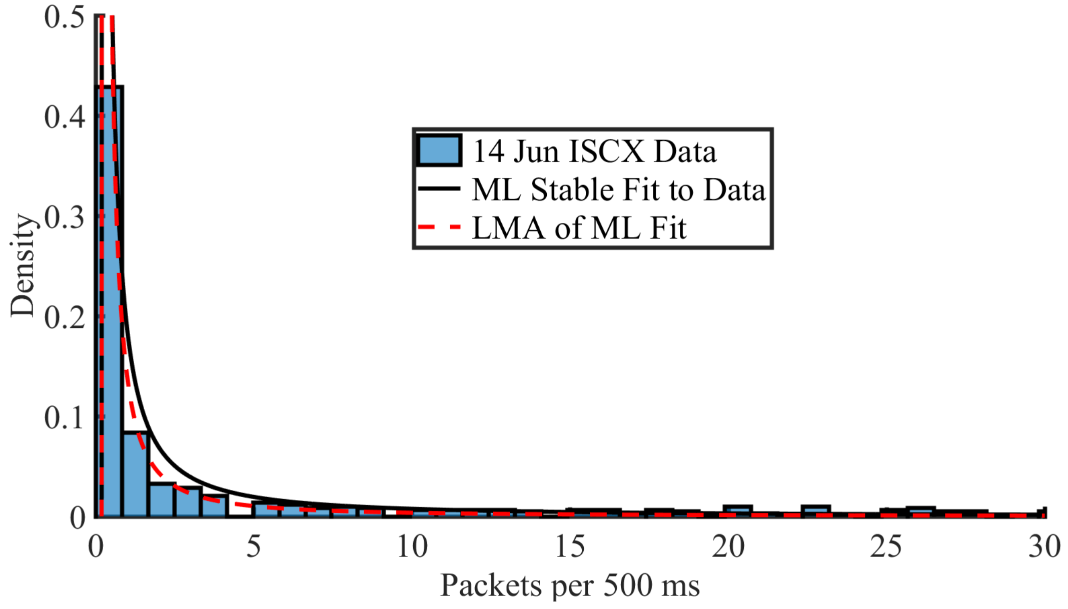


Figure 4.6. End-to-end approximation fit with minimal components. Source: [63].

- Accuracy reduction due to applying a constrained approximation.

To meet our twin goals of reducing accepted errors and enabling real-time implementation, we decided to investigate alternatives that do not require fitting the data with an α-stable distribution. These *stable estimators* are discussed in the next section.

4.4 Approximation versus Estimation

Because stable approximations are only valid in limited cases, we instead choose to model representative attributes of our signals. The SAEs we now discuss are used in test statistics (2.11) that permit estimating single parameters of Z or, more precisely, *representations* of these parameters. These estimators still incur some estimation error, but we do not introduce error through smoothing the data into an α-stable form or add error through approximating the α-stable with finite weighted sums.

Applying these stable *estimators* has the potential to significantly reduce computational costs while still providing accurate characterizations that can be used in a detection system. Single-parameter estimation is computationally-advantageous because in the worst case scenario, we develop the MLE over the allowable space of only a single parameter (vice four). Furthermore, some of these SAEs are non-ML estimators that provide additional savings because their results are obtained directly from $O(N)$-complex manipulations of the sample.

Mathematical characterization of the computational savings for our application is not available in the literature and is a candidate for future work. For this work, we empirically determine the cost savings of our alternative estimation techniques and provide these results in Section 5.6. We now examine the first estimation technique: myriad filtering.

4.4.1 Stable Estimation: The Sample Myriad

The *sample myriad* was first defined in 1996 as an innovation of robust filtering methods developed by Fischer in 1922 [34]. The sample myriad is an adaptive alternative to the median filter, which is itself a robust alternative to the linear (Gaussian) filter. Filters with robust performance are seen as preferred alternatives for recovering signals under non-Gaussian conditions, as it is widely known that the performance of linear filters decline substantially in the presence of non-Gaussian noise [14], [33].

The myriad filter provides the ML (Section 2.3.1) location estimate $\hat{\lambda}$ of a sample using the assumption that the data is stably-distributed [34]. This location is not necessarily the mode or the α-stable location parameter μ (unless less-common definitions of the stable distribution are used [41]), but we have empirically found that it *closely* corresponds to the mode and μ of the best α-stable fit given the distribution of our data from the MAWI traces. For the purposes of this work, the location estimate from the sample myriad (and other estimators discussed in this chapter) is considered the *effective* location of the sample, equivalent to the Gaussian mean μ_2 of a linear process.

Myriad Filter: Derivation

The myriad location is derived from the α-stable special case, closed-form Cauchy distribution. We can rewrite the closed-form definition of the Cauchy distribution (2.5) by substituting the tuning parameter κ for γ and λ for μ as

$$f_Z(z; \lambda, \kappa) = \frac{\kappa}{\pi} \frac{1}{(z - \lambda)^2 + \kappa^2} \tag{4.9}$$

where $\kappa \in (0, \infty)$ is now the scale constant, or dispersion, of the distribution.

To obtain our myriad estimate of location $\hat{\lambda}_M$ (the MLE of λ) given some constant κ, we substitute (4.9) into our ML estimator (2.10) such that

$$\hat{\lambda}_M = \arg\max_{\lambda} \prod_{i=1}^{N} \frac{\kappa}{\pi} \frac{1}{(z_i - \lambda)^2 + \kappa^2}. \tag{4.10}$$

By inspection, maximizing (4.10) is equivalent to minimizing the denominator, thus, (4.10) becomes

$$\hat{\lambda}_M = \arg\min_{\lambda} \prod_{i=1}^{N} \left((z_i - \lambda)^2 + \kappa^2 \right), \tag{4.11}$$

which is defined as the myriad of sample d_i drawn from Z [32]. The myriad location can be seen to be an ML estimate of the location of a Cauchy distribution with spread parameter κ that best fits the data d_i.

Myriad Filter: Application

An intuitive visualization for understanding (4.11) and the effects of κ on $\hat{\lambda}$ is provided in Figure 4.7. In this figure, k is equivalent to κ, and $k_2 >> k_1$. For large k, the location estimate $\hat{\beta}_{k2}$ tends toward the mean of d_i. For the small k, $\hat{\beta}_{k1}$ is drawn to the densest concentration of samples and is not shifted by the outliers on the right.

The concepts illustrated in Figure 4.7 apply to our equations for the myriad as well as subsequent derivations for other tunable estimators of sample location. A small value of the dispersion parameter κ results in a narrow dispersion of the fitted Cauchy distribution, causing the locator to concentrate around the portion of d_i with the heaviest concentration of samples. This leads to a $\hat{\lambda}$ that is resistant to individual outliers and that tends to produce a result very close to the mode of d_i. On the other extreme, as the value of $\kappa \to \infty$, the dispersion of the locator grows and the estimate becomes more sensitive to outliers. For $\kappa = \infty$, it has been shown that $\hat{\lambda} = mean\,(d_i)$ [32].

The implication for our system is that we can adjust, or tune, the value of κ to set the locator behavior we desire in our detection system, e.g., sensitive in some cases, perhaps to provide rapid detection of the onset of an anomaly, and robust when the most accurate $\hat{\lambda}$ is needed

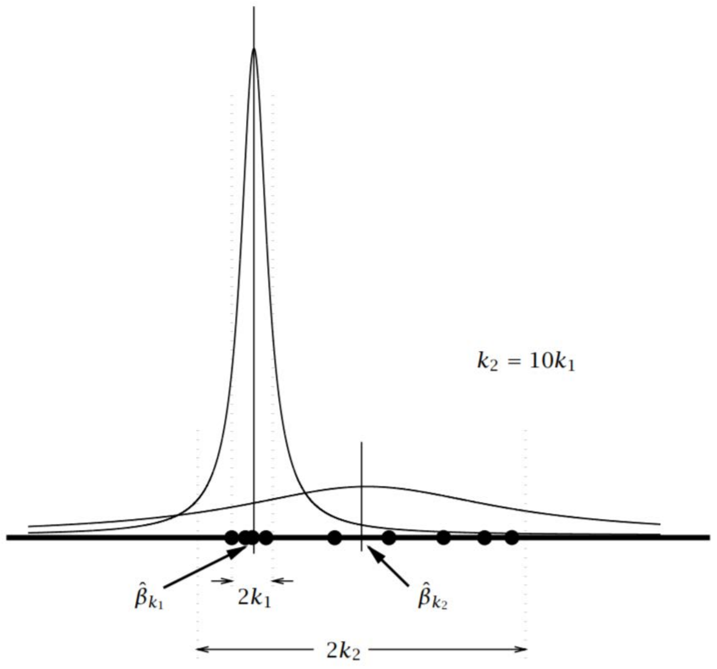


Figure 4.7. Change in Cauchy-based myriad location estimate due to different values of dispersion κ. Source: [67].

to resolve uncertainty.

The myriad filter was derived using an assumption that the input data is symmetric α-stable (SaS) [32] and that the residual errors are Cauchy-distributed. If we instead recognize that our sample is skewed and non-symmetric, it is possible to develop a novel method of estimating λ based on the Lévy distribution.

4.4.2 Lévy Location Estimate

We first derive the LLE following intuitively from the myriad derivation and then examine its implications.

LLE Derivation

Instead of a Cauchy distribution, assume that d_i follows a Lévy distribution (2.4), rewritten using κ for scale and λ for location such that

$$f_Z(z;\kappa,\lambda) = \frac{\kappa}{\sqrt{2\pi}} \frac{1}{(z-\lambda)^{3/2}} e^{-\kappa^2/2(z-\lambda)}. \tag{4.12}$$

Similar to Section 4.4.1, by substituting (4.12) into (2.10), we obtain our LLE $\hat{\lambda}_L$

$$\hat{\lambda}_L = \arg\max_{\lambda} \prod_{i=1}^{N} \left(\frac{\kappa}{\sqrt{2\pi}} \frac{1}{(z_i-\lambda)^{3/2}} e^{-\kappa^2/2(z_i-\lambda)} \right). \tag{4.13}$$

If we again approach (4.13) as a minimization of the non-constant terms in the denominator, we obtain

$$\hat{\lambda}_L = \arg\min_{\lambda} \prod_{i=1}^{N} \left((z_i-\lambda)^{3/2} e^{\kappa^2/2(z_i-\lambda)} \right) \tag{4.14}$$

which, similar to the myriad case, is an ML estimate of the location.

LLE Implications

We continue to investigate the implications of this estimator, which to the best of our knowledge does not exist in the literature. Full examination is an item of future work, but we can draw some initial conclusions from (4.14) and possibly more insight from the alternate LL form given by

$$\hat{\lambda}_L = \arg\min_{\lambda} \sum_{i=1}^{N} \left(\ln(z_i-\lambda)^{3/2} + \frac{\kappa^2}{2(z_i-\lambda)} \right). \tag{4.15}$$

The LLE is similar in form to the myriad and also tunable. As such, it will have a similar computational cost.

For small κ relative to z_i, which we expect for large-magnitude data sets, the right term in (4.15) is dominated by the left. For large κ, the right term dominates. Comparing (4.11) and (4.15), we see that the left terms differ by $1/2\ln(\cdot)$, while the right term is smaller by the inverse of a factor relative to the dispersion of the data $\left((z_i-\lambda)^{-1}\right)$. We expect similar performance to the myriad case but, perhaps, more sensitivity in response to varying values of κ.

Also, like $\hat{\lambda}_M$, LLE is not a selection-based estimator in that (4.15) is evaluated over the range of permissible values of λ and, thus, $\min(d_i) \leq \lambda_{L,M} \leq \max(d_i)$. The ZOL, which we discuss in the following section, *is* a selection-type estimator, and its estimate $\hat{\lambda}_Z$ must equal the value of a repeated sample in d_i [33].

4.4.3 Relationship of Myriad and Levy Location Estimates

Additionally, as the LLE is based on the assumption that the input data is wholly-positive, $\hat{\lambda}_L$ may be more appropriate than the sample myriad for some data sets with extremely heavy tails.

This can be inferred by examining the ratio of the tails as $z \to \infty$. First, consider the Lévy tail from (4.12) where, appropriate for a heavy-tailed positive distribution, we assume $z >> \lambda$. This allows us to neglect λ, yielding

$$f_L(z;\kappa,\lambda) \approx \sqrt{\frac{\kappa}{2\pi}\frac{1}{z^{3/2}}}. \tag{4.16}$$

Similarly, the Cauchy tail is

$$f_C(z;\kappa,\lambda) \approx \frac{\kappa}{\kappa^2 + z^2}. \tag{4.17}$$

Taking the ratio of the tails leads to

$$\frac{f_L(\cdot)}{f_C(\cdot)} \approx \frac{\sqrt{\kappa}}{2\pi}\frac{1}{\kappa}\frac{(\kappa^2 + z^2)}{z^{3/2}}. \tag{4.18}$$

Simplifying, neglecting small constants, and assuming for practical applications $\kappa << z$ (but not negligible), we obtain the asymptotic proportionality relationship of

$$\frac{f_L()}{f_C()} \approx \frac{\sqrt{z}}{\sqrt{\kappa}}, \tag{4.19}$$

which demonstrates that the Lévy tail is larger than the Cauchy tail and, as such, *may* be able to provide a better location estimate for extremely heavy-tailed data (i.e., $\alpha \in (0, 0.5)$).

Empirical evaluation to this point has validated that the performance of the LLE is very similar to $\hat{\lambda}_M$ (as discussed in Section 4.6). The LLE produces results nearly identical to the myriad and very similar to our other location estimator, ZOL, in more than 500 evaluated cases. As previously discussed, a full performance evaluation and additional application of the LLE is one of our items of future work. Preliminary comparisons of the outputs of all location estimates are discussed in Section 4.6 after we derive additional estimators from the family of ZOS.

4.5 Stable Estimation: Zero-Order Statistics

ZOS were developed subsequent to the sample myriad by the same research group as a way of defining equivalents to measures of second-order (Gaussian) processes [33]. As opposed to $\hat{\lambda}_L$ and $\hat{\lambda}_M$, ZOS estimators are derived from non-special cases of the stable family. ZOS are also a more flexible, stable-based alternative to fractional lower-order statistics (FLOS) which have a proven history of success in non-Gaussian environments.

FLOS generally assume data follows a symmetric α-stable distribution and can only be determined for moments $p < \alpha$. In the case of very small α, a FLOS estimate may not exist [33]. Because we do not wish to constrain our methodology based on the value of α, we chose not to apply FLOS estimates in our work. Also, our data is not symmetric, and while SaS approximations to skewed variables exist and could be used to enable FLOS-based estimation, this approach requires additional computational cost (and approximation error) similar to the mixtures examined in Section 4.1 [68].

ZOS, on the other hand, provide second-order equivalent representations of power, dispersion, and location while making no constraining assumptions regarding the input signal. We begin our discussion with the power estimation method, or zero-order power (ZOP), comparing it with traditional second-order power, then derive the location and dispersion estimates.

4.5.1 Zero-Order Power

Second-order processes use power $\mathbb{E}(Z)^2$ as an expression of the strength of a process. Because of the MOC of Section 2.2, an alternative definition of power must be determined if we wish to characterize the strength of stable processes over the permissible range of $\alpha \in (0, 2]$.

To nullify the MOC, ZOS uses the fact the logarithm of an α-stable distribution has finite first and second-order moments as $\mathbb{E}(\ln |X|) < \infty$ [33]. These *logarithmic moments* enable defining ZOP as well as ZOS estimates of dispersion (ZOD) and location (ZOL).

Zero-Order Power: Definition and Estimation

ZOP [33] is defined as

$$\varphi_0(Z) = e^{\mathbb{E}(\ln |Z|)}. \tag{4.20}$$

The literature also uses the term geometric power for ZOP, and its estimate is equivalent to the geometric mean of a positive data sample [33]. This is demonstrated as we consider the problem of easily determining ZOP for stable distributions, which do not have a closed form.

To estimate ZOP [23], we replace the expectation operator in (4.20) with its discrete analogue such that

$$\hat{\varphi}_0(X) = exp\left(\frac{1}{N}\sum_{i=1}^{N} ln|z_i|\right). \tag{4.21}$$

Rearranging these terms gives

$$\hat{\varphi}_0(X) = exp\left(\sum_{i=1}^{N} ln|z_i|\right)^{\frac{1}{N}} = \left(\prod_{i=1}^{N} |z_i|\right)^{\frac{1}{N}} \tag{4.22}$$

the geometric mean of $|z_i|$ [23].

Zero-Order Power: Analysis
In the limit, ZOP is equivalent to FLOS moments p such that

$$\varphi_0 = \lim_{p \to 0} \varphi_p. \tag{4.23}$$

The proof of (4.23) is available [23], though not essential for our purposes. The asymptotic equivalence of (4.23) is important because of the long history of FLOS and its examination and successful application to detection in the fields of underwater acoustics and radar [14], [43].

Returning to (4.22), we know that the geometric mean is both scale invariant and an ML estimator of ϕ_0 if d_i is Pareto-distributed [33]; thus, while ZOP is not strictly derived using α-stable assumptions, it is firmly grounded in heavy-tailed and FLOS estimation and should perform robustly in non-Gaussian environments. Also of particular interest for our purposes, ZOP can be estimated from the data and does not incur the computational costs of distribution fitting or numerical integration.

ZOP has the following additional properties important to our uses:

1. ZOP is a *scale parameter* and, thus, an indicator of the dispersion of a sample.
2. ZOP is an indicator of process strength proportional to the magnitude of the sample data.
3. ZOP is multiplicative such that $\varphi_0(XY) = \varphi_0(X)\varphi_0(Y)$

We refer the interested reader to the proofs of these properties [23]; they are not essential for our purposes.

Similar to ZOP, ZOD can be estimated directly from data at low cost. The derivation of ZOD, however, uses the definition of ZOL, so we will develop ZOL.

4.5.2 Zero-Order Location

Gonzalez defines ZOL as the stable equivalent of the Gaussian mean, and it is the only non-ML location estimation approach we apply [33].

Zero-Order Location: Derivation

The derivation of ZOL begins with the definition of the linear sample location (or mean μ_2), which can be defined as the location value which, for all possible shifts μ, minimizes the power of a shifted variable Z [69]. Equivalently,

$$\mu_2 = \arg\min_{\mu} \mathbb{E}(Z-\mu)^2. \tag{4.24}$$

The right side of (4.24) can be seen to be equivalent to second-order power φ_2 for a shifted process $(Z-\mu)$ such that

$$\mu_2 = \arg\min_{\mu} \varphi_2(Z-\mu). \tag{4.25}$$

Similar to the second-order definition of location, ZOL is defined as the value λ_0 that minimizes the power of the shifted process [33]. We can reformulate (4.25) with zero-order equivalents of power φ_0 and location λ_0 to obtain

$$\lambda_0 = \arg\min_{\lambda} \varphi_0(Z-\lambda). \tag{4.26}$$

Replacing φ_0 with its definition from (4.20) yields

$$\lambda_0 = \arg\min_{\lambda} e^{(\mathbb{E}\ln|Z-\lambda|)} = \arg\min_{\lambda} \mathbb{E}|Z-\lambda|. \tag{4.27}$$

We wish to generalize (4.27) to enable its application to our problem. To do so, we introduce our theorem regarding stably-distributed network traffic. This enables our estimate of ZOL, $\hat{\lambda}_0$, as well as subsequent development of a detector threshold algorithm.

Theorem 1: *Even when network traffic is stably-distributed, samples of network traffic are finite and, thus, have finite first and second-order moments.*

Proof: The proof follows trivially from Lemma 1.

Lemma 1.1: *Any sample $\boldsymbol{d}$ of network traffic is finite, of size proportional to the sample period ϖ and traffic rate R. As such, it is possible to calculate the mean $\mathbb{E}(\boldsymbol{d})$ and second moment $\mathbb{E}(\boldsymbol{d})^2$ of any numerical representation of the features of this network traffic sample.*

While it is possibly to apply a stable distribution (which does not necessarily have a first moment) to model aspects of network traffic, the actual network traffic samples which define the model are, by definition, finite with first-order moments.

Lemma 1.1 provides a theoretical foundation for estimating ZOL, as we can discretize the expectation in (4.27) such that

$$\hat{\lambda}_0 = \arg\min_{\lambda} \frac{1}{N} \sum_{i=1}^{N} |z_i - \lambda|, \tag{4.28}$$

where $\boldsymbol{d} = \sum_N (z_i)$.

Dropping the normalization term $1/N$ and taking the natural logarithm of each term to compact the sum does not change the resulting estimate [33], leading to

$$\hat{\lambda}_0 = \arg\min_{\lambda} \sum_{i=1}^{N} \ln |z_i - \lambda|. \tag{4.29}$$

There is one issue with this formula: Ties between estimates will be produced whenever $z_i = \lambda$ because the minimization argument returns a value of $-\infty$ [33]. These ties can be avoided by creating a compact set of potential estimates ($\Omega_{\hat{\lambda}}$) which exclude ties and constraining $\lambda \in \Omega_{\hat{\lambda}}$ [33].

This set, and subsequent evaluation of (4.29) at all possible values of $\lambda \rightarrow z_i$, transforms the ZOL estimator to the form of

$$\hat{\lambda}_0 = \arg\min_{z_j \in \xi} \sum_{i=1, z_i \neq z_j}^{N} \ln |z_i - z_j|, \tag{4.30}$$

where ξ is the set of repeated values in a given observation of Z [33]. The proof of this transformation is not essential to our work but is available [33].

Note that this form permits estimation of λ_0 directly from data and *without* having to search all possible values of location, reducing computational costs. ZOL has one drawback in that small data sets where only outlying values are repeated are mis-estimated by $\hat{\lambda}_0$. In our application, data sets generally have a well-defined mode, fairly compact range, and a large number of samples (on the order of 750 - 2000+). Our conclusion, based on analysis of the risk and empirical evaluation, is that the ZOL is not eliminated as a potential estimator but should be applied with an element of caution.

Now that we have defined both λ_0 and $\hat{\lambda}_0$, we can use these definitions to develop the last of our ZOS-based estimators, ZOD.

4.5.3 Zero-Order Dispersion

We develop our definition of ZOD δ_0, equivalent to second-order variance, similar to our methodology for ZOP. Returning to the definition of the mean in (4.25), where

$$\mu_2 = \arg\min_{\mu} \varphi_2(Z - \mu), \tag{4.31}$$

we can use the minimum *result* μ_2 of this definition. Given that $\varphi = \mathbb{E}(Z)^2$, the operand of (4.31) can be seen to be the second-central moment, or variance σ_2^2, of Z. The variance of Z can, thus, be written as

$$\sigma_2^2 = \min_{\mu} \varphi_2(Z - \mu_2). \tag{4.32}$$

We use this to define the zero-order equivalent of variance by replacing second-order terms with their ZOS equivalents [23] such that

$$\delta_0 = \min_{\lambda} \varphi_0(Z - \lambda_0). \tag{4.33}$$

Using the previous definition of ZOL as the actual value that minimizes the shifted process (4.29) [23], we estimate (4.33) as

$$\delta_0 = \varphi_0(Z - \hat{\lambda}_0). \tag{4.34}$$

Little practical application of ZOD exists in the literature. It is, though, a useful measure of the spread of α-stable data, particularly because it can also be estimated without incurring the costs of curve fitting. Our application and assessment of the utility of ZOD is contained in Chapter 5; we now assess the effectiveness of the various estimators derived in this chapter.

4.6 Location Estimator Accuracy

Preliminary results comparing estimator performance are shown in Table 4.2. In this table, the attack and benign location estimation results for 1,798 individual windows of the MAWI 20160428 trace are summarized. The stable location parameter μ for each sample was obtained from the ML α-stable fit to each sample using STABLE [13]. The sample mode was obtained by numerical search of the sample data vice the histogram.

The results in Table 4.2 warrant further discussion. First, the location estimates in many (but not all) cases are heavy-tailed, hence the sometimes-large divergence between the mean and median differences. Second, the LLE is the most accurate estimator of stable location μ. The ZOL and, to an extent, the sample myriad results correspond more closely to the sample mode. This is expected according to the theoretical design of the ZOL as a selection-type estimator; this is also the expectation from the literature that as tuning parameter $\kappa \to 0$, the sample myriad behaves as *mode-type* estimator [34].

Second, all of the stable-based methods outperform the median and mean under benign conditions by at least 50 percent. Under attack conditions, when the data is frequently near-Gaussian, the mean and median can be seen to be the best stable *location* estimators of μ but less accurate than the alternate methods in determining the sample mode. Overall, the results of this section imply that the best estimator of the *mode* of stably-distributed data is the ZOL, while the best estimate of the stable location is provided by the LLE in the benign case and the LLE or the mean, median, or LLE in the attack case.

Finally, obtaining representative estimates directly from data is advantageous because it avoids the computational costs of curve fitting to estimate the four α-stable parameters. Highly-efficient MLE-based algorithms are available to rapidly perform this estimation, but the empirical results in Section 5.6 indicate that distribution fitting is approximately 25 times more costly than application of any of the location estimates discussed above, so our detection system will use the estimators in this chapter in order to support real-time application.

Table 4.2. Performance comparison of the sample myriad λ_M, ZOL λ_0, and LLE λ_L

	Attack		Benign	
Difference	Median	Mean	Median	Mean
$\hat{\lambda}_M - \mu$	13.1	13.2	1.7	1.9
$\hat{\lambda}_0 - \mu$	12.0	12.5	1.8	2.0
$\hat{\lambda}_L - \mu$	7.0	10.0	0.2	0.1
Median||Mean - μ	5.8	3.8	6.2	14.6
$\hat{\lambda}_M$−mode	0	12.6	10.0	10.5
$\hat{\lambda}_0$−mode	1.0	13.2	1.8	2.0
$\hat{\lambda}_L$− mode	10.1	15.7	12	12.3
Median||Mean - mode	20.0	21.9	19.0	27.4
Mean Data Range	635		468	
Windows	767		1031	
Windows with $\alpha \geq 1.9$	60.2%		8.5%	

The conclusions of this section must be validated through similar analysis using additional data sets, additional values of κ_L and κ_M, and with data characterized by a wider range of α (particularly $\alpha < 1.5$). This is an item of future work; however, for any stably-distributed network data with $\alpha \geq 1.5$ and similar values of κ, it is likely that any change of data set will yield similar results.

Overall, these preliminary results imply that if we wish to utilize a location-based approach in our detection system, we should utilize ZOL as it is the most accurate mode estimator under all conditions (keeping in mind the small-sample limitations discussed in Section 4.5.2).

4.7 Summary

In this chapter, we examined the feasibility of modeling our α-stable network traffic data using both closed-form solutions based on hypergeometric solutions and approximations based on stable special cases. Both approaches use weighted mixtures of closed-form distributions to harness the accuracy of the α-stable distribution; however, both approaches suffer increased computational cost through first estimating the best α-stable fit through

MLE and also are constrained in the range of α or β values they can model.

In a third, more lightweight approach, we examined representative attributes that could describe the best α-stable fit to the data and that could be calculated, rather than estimated, directly from the data. These SAEs are similar to Gaussian estimates of location, dispersion, and power. Heavy-tailed assumptions were used to develop estimators for α-stable location, dispersion, and power attributes. Two of these location estimators, the myriad and LLE, were shown to be ML approaches derived using the Cauchy and Lévy assumptions, respectively, for the distribution of the data. ZOP was defined as an MLE of the second-order moment using Pareto assumptions. ZOL is not an MLE approach, and there is a small data set disadvantage that must be considered when applying ZOL, but it has been empirically shown to produce the most accurate estimates of mode in both attack and benign conditions. Portions of results and conclusions from these analyses were presented in Section 4.6, with the remainder held in Chapter 5.

CHAPTER 5: Statistical Detection for α-Stable Data

Thus far, we have demonstrated that many cases of network traffic are stably-distributed. We have also reviewed detection implications of the α-stable distribution and developed estimation techniques suited for α-stable data.

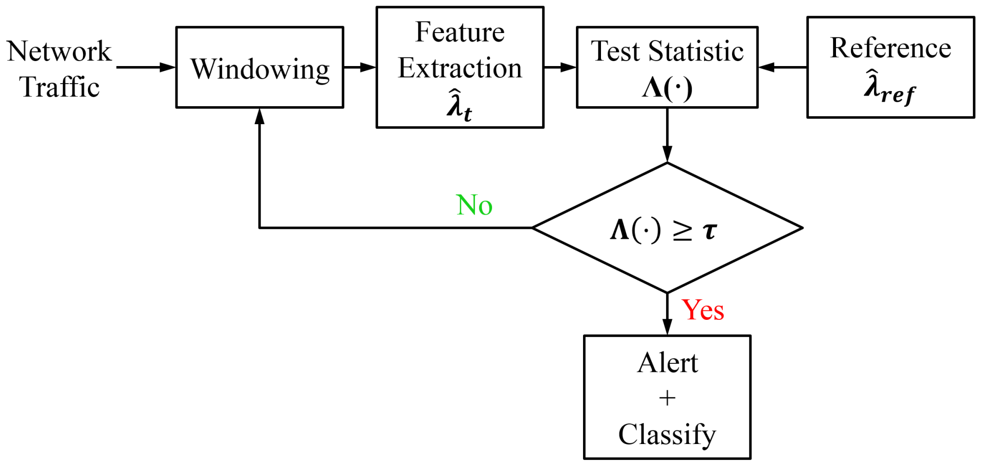
In this chapter, we develop the detector, processes, and test statistics for our anomaly detection system. To describe our detection environment and choose our input signals, we first develop a modular detection system model. Using this model, we examine a suitable statistical test (the GLRT) and then adapt a generic GLRT to our specific inputs. Any test requires a threshold estimate suitable for α-stable data; this threshold relationship is also derived. The developed algorithms and processes yield a detection system that responds to the selected signal features calculated using the α-stable estimators from Chapter 4. The end result is an ML, or in certain cases pseudo-ML, GLRT-based detection system for detecting volumetric anomalies in network traffic due to DoS attacks.

5.1 Detection Model

Developing a well-defined system model fully enables understanding data flows and performance influences and ultimately enables implementation of a detection system. We began by reviewing the literature for a state-of-the-practice model that can serve as a starting point for α-stable innovation.

Unfortunately, the existing statistical detection literature does not provide starting designs with sufficient detail for our purposes. Accordingly, we used the literature as a starting point to develop our own model of a modular detection system, shown in Figure 5.1.

This development process increased our comprehension of the impacts of design choices and enabled us to identify specific processes where non-Gaussian methods might be applied to improve system performance. A basic model of our system is shown in Figure 5.1; a more detailed system model is contained in the Appendix. Aspects of the basic model warrant discussion in further detail; we begin with an overview of the system functions,


Network Traffic
Windowing
Feature Extraction $\hat{\boldsymbol{\lambda}}_t$
Test Statistic $\boldsymbol{\Lambda}(\cdot)$
Reference $\hat{\boldsymbol{\lambda}}_{ref}$
$\boldsymbol{\Lambda}(\cdot) \geq \boldsymbol{\tau}$
No
Yes
Alert + Classify
Figure 5.1. Network anomaly detection system model used in this research

then examine essential processes in greater detail.

Our anomaly detection (AD) system collects network traffic for a given period of time, the sample *window*, ϖ. To eliminate unnecessary data in the sample, the system filters the collected data for specific features that can be analyzed using the tests in the detector. Commonly-analyzed features include source and destination IP addresses, packet rate, and byte rate.

Some aspect of the chosen feature is then measured (i.e., counted) over a uniform counting period, the sub-window (Δ_{sw}). Each measurement per counting period forms an individual vector element d_i; concatenating M instances of d_i produces the input *feature vector* $\boldsymbol{d}_t$. The detection process then extracts pertinent attributes or attribute estimates from the vector to produce the current-window input to the test statistic $\hat{\lambda}_t$.

The *test statistic* compares $\hat{\lambda}_t$ to an estimate of the feature's normal, or expected, value $\hat{\lambda}_{ref}$. In a more advanced implementation (e.g., the modular system in the Appendix), this estimate of normal is produced through a combination of memory, weighting (or decay), and updating processes. In our proof-of-concept system, the normal estimate is simply the most recent prior $\boldsymbol{d}$ that does not contain attack traffic. We call this the *benign prior reference sample* $\boldsymbol{d}_{ref}$.

A *score* is then produced by comparing $\boldsymbol{d}_{ref}$ to $\boldsymbol{d}_t$ based on a test statistic that is appropriately designed for the selected feature. If the score exceeds the *threshold*, an alert is generated through a reporting process.

Optionally, a more complex adaptive implementation, such as that shown in the Appendix, uses a mechanism to control and update system settings such as the monitored features, window size, and threshold. This work does not add those layers of complexity; adaptive improvements are items of future work.

Now that we have described an overall framework for processing and testing, more specific descriptions are necessary for some of the key processes in Figure 5.1.

5.2 Key Detection Processes

Some detection system processes are relatively straightforward and not worth additional discussion outside of impacts on accuracy or documentation outside of the code in the Appendix (e.g., filtering and feature extraction). Others bear elaboration, namely windowing and feature selection, which are discussed in further detail in this section, and *activity recognition*, a complex process that is rigorously described in Section 5.3. Once we describe the processes for measuring the contents of the signal, it is then defined and tested.

5.2.1 Data Windowing

As we demonstrated in Chapter 3, the size of the window and the sub-window can notably affect the fit of the model and distribution of features and, thus, can influence the performance of the entire detection system.

In addition to ϖ and Δ_{sw}, window overlap Δ_o, defined in percent, is another windowing concept that affects the feature vector. This term can be varied; large overlaps divide changes due to anomalies over multiple $\boldsymbol{d}_t$, which can make detecting a specific change more difficult (particularly with transient anomalies with a duration approaching ϖ).

To maximize the accuracy of the Monte Carlo simulations, given the relatively small ($<$ 15 minute) data traces that were available, the results examined in this work generally set $\Delta_o \in [50\%, 90\%]$. Examination and optimization of the effects of Δ_o on system accuracy is an item for future work, but some qualitative examination of the effects of large Δ_o is contained in Section 5.5.

5.2.2 Feature Vector

Each sample is a feature vector; the detection system can test one or more feature vectors in each window using one or more test statistics. A multiple-test implementation generates an ensemble score from multiple test statistics and is called a *layered* detection system [70]. Layering can be used to refine the results of individual test statistics and has the potential to produce more accurate detection systems. Layering and ensemble techniques are a highly interesting item of future work, as they seem to be relatively unexamined in the recent literature and may present significant opportunity for accuracy improvement.

In keeping with a proof-of-concept demonstration, however, in this work we examine only one feature (packets per sub-window) over one window and apply only one test to the feature vector; thus, our detector ultimately compares a $\boldsymbol{d_t}$ and $\boldsymbol{d_{ref}}$, comprised of M counts of packets over time ϖ. With the sampling process fully described, let us now define our signals and determine our hypotheses for testing.

5.2.3 Signal Definitions

As outlined in Chapter 1, our problem is that of detecting a change in streaming data represented as discrete time series. This problem is complicated by the following unknowns:

1. The distribution of the signal varies in each sample $\boldsymbol{d_t}$.
2. The time of change is unknown; each $\boldsymbol{d_t}$ of the series must be examined.

Our signal is constantly changing due to small random effects, as demonstrated by Figure 3.5. The signal *significantly* changes at the onset of an attack, and these changes persist for the duration of the attack, as demonstrated in Chapter 3. In this chapter, we also showed that the signal, both before and after the change, can be most accurately (and flexibly) modeled using RVs following an α-stable distribution.

The distribution of the signal both before and after the attack is unknown and, should we wish to apply parametric methods, must be estimated for each $\boldsymbol{d_t}$ of our received signal, Z. The vector $\boldsymbol{\theta}$ was defined in Chapter 4 as the set of stable distribution parameters $\{\alpha, \beta, \gamma, \mu\}$, where $\boldsymbol{\theta_b}$ and $\boldsymbol{\theta_k}$ describe received benign and attack traffic, respectively.

In the benign case of the signal without an attack

$$\boldsymbol{d}_t \sim X \sim S(\boldsymbol{\theta_b}) \tag{5.1}$$

where RV X describes benign, but constantly-changing, background traffic in window t, collected via an ordered time series of samples.

Due to the properties of random variables, adding attack traffic distributed per RV K results in convolution of the benign and attack traffic, changing the distribution of the received signal such that

$$\boldsymbol{d}_t \sim X * K \sim S(\boldsymbol{\theta_k}) \tag{5.2}$$

These definitions of our received signals can now be used to specify hypotheses that differentiate between the following cases:

$\mathcal{H}_0$: $\boldsymbol{d}_t \sim S(\boldsymbol{\theta_b})$ and
$\mathcal{H}_1$: $\boldsymbol{d}_t \sim S(\boldsymbol{\theta_k})$.

With the overall system specified and our hypotheses described, we now proceed to defining our testing process. This is the objective of the next section.

5.3 Activity Recognition: The Detector

The process of Activity Recognition, where the test statistic is compared to the detection threshold, is critical to system accuracy and bears examination in detail. We begin this discussion by developing the actual detector test(s) and then determine an approach for calculating the threshold to which the test results are compared.

5.3.1 Testing Approach of Prior Work

Prior α-stable network anomaly detection work used a GLRT to perform parametric hypothesis testing, comparing traffic samples to a historical library of known benign traffic samples [10]. This approach borders on a signature-type methodology, where accurate representative samples of benign traffic are required for the test to perform. Signature methods are known to be ineffective against unknown attacks. Additionally, the previous work's accuracy can be reduced should current traffic not follow previous patterns (e.g., due to the installation of new devices in the case of a small network, or a change in peering agreements in the case of an autonomous system).

To support real-time implementation and reduce system complexity, our implementation uses non-parametric data transformations (i.e., SAEs) to produce single-valued estimates. If we wish to the avoid the high costs of estimating $\boldsymbol{\theta}$, then our methods must be non-parametric, which we achieve by applying estimators that function with data of any distribution (though they are formulated for our anticipated distribution). Similarly, we apply a non-Bayesian GLRT to compare the signal and associated hypotheses of Section 5.2.3. The non-linear nature of our signal requires use of a likelihood ratio test and the Bayesian-based approaches are computationally costly and require a priori assumptions that may be incorrect [49].

To the best of our knowledge, this work constitutes the first real-time, non-parametric network anomaly detection system developed using α-stable approaches. Our non-parametric methods feed the lightweight location and dispersion estimates of Chapter 4 into a GLRT, which are now derived.

5.3.2 Development of the GLRT

If the distribution of the signal was known beforehand, we would apply the optimal Neyman-Pearson test [52]. For a constrained FAR χ, this optimal test is given by

$$L(Z) = \frac{p\,(\boldsymbol{d}_t; \mathcal{H}_1)}{p(\boldsymbol{d}_t; \mathcal{H}_0)} > \tau_{NP}, \tag{5.3}$$

where τ is given by

$$P_{fa} = P\big[L(\boldsymbol{d}_t) > \tau_{NP}; \mathcal{H}_0\big] = \chi. \tag{5.4}$$

As the attack signal is unknown, we instead apply a GLRT in our detector. The GLRT $\Lambda(\cdot)$ adapts the optimal likelihood ratio test of (5.3) for the unknown parameters $\boldsymbol{\theta}$ in our signal $\boldsymbol{d}_t$. It is the most-frequently applied alternative to a Neyman-Pearson test as it has been proven to approach optimality for large data records or signal-to-noise ratios (SNRs) [52].

Our GLRT compares the likelihood of our two hypotheses $p(\boldsymbol{d}_t;\mathcal{H}_1)$ and $p(\boldsymbol{d}_t;\mathcal{H}_0)$ to a threshold τ, deciding $\mathcal{H}_1$ if

$$\Lambda(\boldsymbol{d}_t) = \frac{p\left(\boldsymbol{d}_t;\mathcal{H}_1\right)}{p(\boldsymbol{d}_t;\mathcal{H}_0)} > \tau \tag{5.5}$$

and H_0 if $\Lambda(\boldsymbol{d}_t) \leq \tau$ [52]. The threshold τ is set empirically or derived from the fixed FAR χ by solving

$$P_{fa} = P\big[\Lambda(\boldsymbol{d}_t) > \tau;\mathcal{H}_0\big] = \chi. \tag{5.6}$$

Evaluation of this likelihood ratio as written requires obtaining the MLE of the two hypotheses under consideration. MLEs are computationally costly for stable distributions because four related parameters must be determined; empirical observation has shown that estimation of the four stable function parameters for a single data sample (i.e., fitting a stable PDF to the sample) is approximately 10-500x more expensive than applying the SAEs. Accordingly, while an anomaly detection system *could* be derived from MLEs of the data (and is a candidate for future work), we maximize the scalability of our system by using the more lightweight SAEs developed in Chapter 4.

Stable Attribute Estimator-based GLRTs

To do so, (5.5) must be adapted to our problem of SAE-based detection in a time series. This GLRT can be re-written to use the current and benign reference samples in the form of

$$\Lambda(\boldsymbol{d}_t) = \frac{p\left(\boldsymbol{d}_t\right)}{p\left(\boldsymbol{d}_{ref}\right)} \gtrless \tau \tag{5.7}$$

where $\boldsymbol{d}_t \sim Z_t$ (the current process sample), and $\boldsymbol{d}_{ref} \sim Z_{ref}$ for the most recent, benign-traffic-only sample (i.e., $\mathcal{H}_0$).

In our lightweight approach, the numerator and denominator are replaced by any suitable SAE (we assume $\hat{\lambda}_0$ for now) such that (5.7) assumes the form applied in our detection system,

$$\Lambda(\boldsymbol{d}_t) = \frac{\hat{\lambda}_{0,t}}{\hat{\lambda}_{0,ref}} \gtrless \tau \tag{5.8}$$

or alternately, applying the $\ln(\cdot)$ operator

$$\ln \Lambda(\boldsymbol{d}_t) = \ln\left(\hat{\lambda}_{0,t} - \hat{\lambda}_{0,ref}\right) \gtrless \ln \tau. \tag{5.9}$$

We have, thus, developed a log-likelihood ratio test statistic for our detection system that can be expressed in closed form and is less computationally expensive than distribution-based MLE approaches, but while we now have a form for the left side of test statistic inequality, the threshold remains to be defined.

5.3.3 Threshold

The literature frequently defines τ by evaluating the PDF of the signal with respect to a fixed FAR χ; this is the CFAR approach frequently used in radar and other detection systems [43], [52], [71]. This is a viable method for our implementation, and the development of this test is fairly straightforward based on the null hypothesis: numerically integrate a benign reference sample ($\mathcal{H}_0$) to determine the threshold τ, which includes $(1 - \chi)$ percent of samples.

As an alternative, τ can be estimated from previous samples by a variety of methods suited for bounding RVs. Unfortunately, the vast majority of threshold determination approaches in the literature rely on assumptions of Gaussianity. Finding a suitable non-Gaussian approach required a search of robust techniques and outlier analysis methods for alternate approaches, such as non-parametric Markov and Chebyshev bounds. Non-parametric approaches have increased applicability to various types of data and, as a consequence, looser bounds [60].

Our literature review identified a promising candidate, the Hoeffding inequality, which is derived using Markov's inequality and has been shown to produce tighter bounds than both the Markov and Chebyshev inequalities [60]. The Hoeffding bound is also non-parametric and advantageous for our application because it has a more general (i.e., non-exponential) tail assumption than the other non-parametric candidates [60].

For an RV X that can be expressed as the sum of N independent RVs [60], the Hoeffding inequality defines an upper bound ρ such that

$$P(X - \mathbb{E}[X] > \rho) \leq \exp\left(\frac{-2\rho^2}{\sum_{i=1}^{N}(u_i - l_i)^2}\right) \tag{5.10}$$

where u_i and l_i are the upper and lower bounds of the independent RVs. The proof is available [60], but for our purposes the constraints of the Hoeffding Inequality are more important.

The Hoeffding Inequality assumes *bounded* RVs and is defined using the first moment; α-stable distributions with $\alpha < 1$ do not possess either of these properties. Rather than constrain its applicability in our detection system to samples where $\alpha > 1$, we make use of Theorem 1 (Section 4.5) to permit application of finite measures of finite data that follows a heavy-tailed distribution.

Lemma 1.1 enables the application of the Hoeffding bound in our detection system. For chosen FAR χ and $N = 1$ samples, (5.10) can be reformulated to define our threshold

$$\tau = \rho \leq \sqrt{\frac{\ln(\chi)(u_i - l_i)^2}{-2}}. \tag{5.11}$$

We have now defined the hypothesis test, test statistics, and threshold that are applied in the Activity Recognition module; however, one additional sub-process that was not explicitly included in Figure 5.1 bears discussion, differencing.

5.3.4 Differencing

Differencing is a data transformation accomplished by subtracting a prior sample from the current sample. This technique is frequently applied in time-series analysis to mitigate the effects of non-stationary trends in the data, but we do not use it for this purpose as we have previously established that our data is trend stationary over the utilized sampling periods. More aligned to our objectives, differencing is also used to mitigate the impacts of random walks or other short-term cyclic influences that do not affect the series stationarity [62], such as those seen in Figure 3.5.

Differencing Demonstration

We apply differencing in some of our detector tests for two reasons. First, as our data is subject to random walks, it may be logical to remove this potential accuracy degradation if sufficiently small Δ_o is used. Second, as demonstrated in Chapter 3, DoS attacks broaden the distribution of the traffic sample by causing a higher per-sub-window packet rate over the baseline benign traffic. Differencing the sample under test with a prior benign sample increases the apparent signal strength of the attack by more sharply revealing the outliers.

This can best be demonstrated visually and is illustrated in Figure 5.2. This figure was created by randomly choosing two attack and three benign windows from the MAWI 20160428 cases, then differencing each sample with a benign sample and plotting the resulting histograms. The outliers in the attack cases show how differencing more distinctly reveals changes in the traffic distribution during an attack.

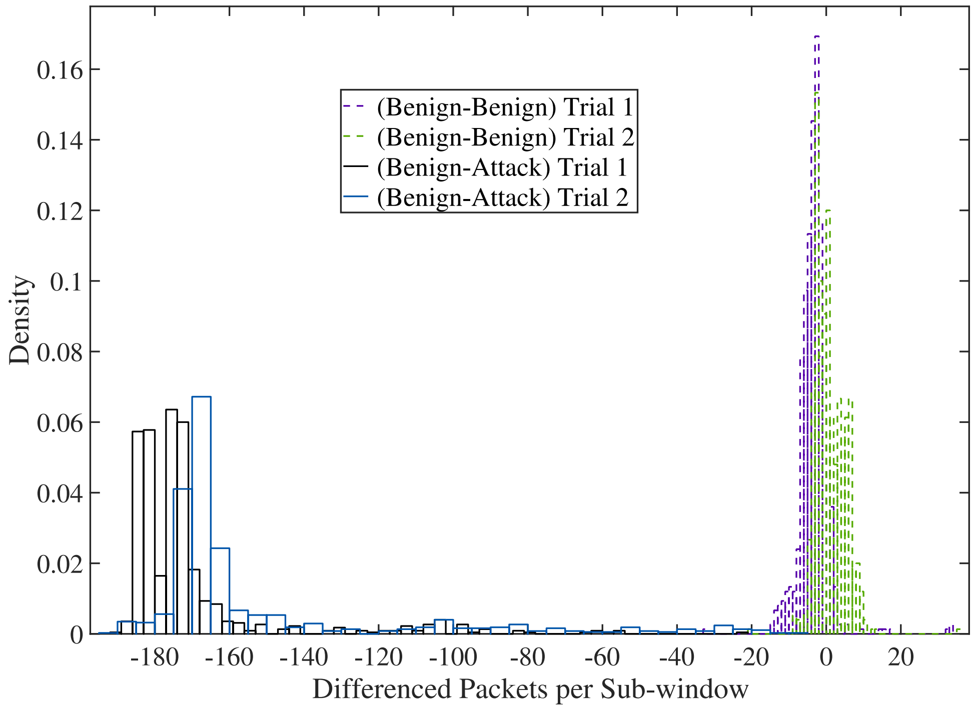


Figure 5.2. Varying effect of differencing benign and attack cases

Examining this figure, we see that differencing tends to eliminate "redundant" measurements that occur in both samples and add little value when determining the difference between samples. The high similarity of the benign samples is proven by the clustering of the differenced results around zero. Differencing the attack samples, however, reveals large-magnitude outliers concentrated far from the origin. This result should produce a strong signal, as measured by one or more of our SAEs, and large separation from the benign case, leading to a large P_d at relatively small P_{fa}.

Differencing Effect on Test Statistic

When differencing is applied in our detection system, we subtract the immediately-prior benign window. In practicality, this results in our GLRT test of (5.8) becoming a differencing GLRT $\Lambda_d(\cdot)$, expressed as

$$\Lambda_d(\boldsymbol{d}_t) = \frac{\hat{\delta}_{0,t} - \hat{\delta}_{0,ref}}{\hat{\delta}_{0,ref} - \hat{\delta}_{0(ref-1)}} \gtrless \tau. \tag{5.12}$$

Again, any of our SAEs can be substituted for the estimator in the numerator and denominator in (5.12).

5.4 Summary of Detection Processes

To this point in the chapter, we have defined the inputs and the essential internal processes of our detection system through developing and elaborating our system model. Key relationships were derived for the detector, including the GLRT test statistics $\Lambda(\cdot)$ and the associated threshold τ. The test statistics of (5.8) or (5.12) can be used with any SAE; these estimators of $\boldsymbol{d}_t$ are summarized in Table 5.1.

Table 5.1. Stable attributes using stable-based estimators

Description	Symbol	Formula
Myriad Location	$\hat{\lambda}_M$	$\underset{\lambda}{\arg\min} \prod_{i=1}^{N} \left(\kappa^2 + (z_i - \lambda)^2\right)$
Levy Location	$\hat{\lambda}_L$	$\underset{\lambda}{\arg\min} \prod_{i=1}^{N} \left((z_i - \lambda)^{3/2} e^{\kappa^2/2(z_i-\lambda)}\right)$
Zero Order Location (ZOL)	$\hat{\lambda}_0$	$\arg\min \sum_N \ln \lvert y_i - \delta \rvert$
Zero Order Dispersion (ZOD)	δ_0	$\hat{\varphi}_0(y_i - \hat{\delta}_0)$
Zero Order Power (ZOP)	$\hat{\varphi}$	$\exp(\frac{1}{N} \sum_N \ln \lvert y_i \rvert)$

Definition of the test statistics allows comparing performance results between SAEs as well as against classical tests such as mean, median, and variance, and this is the subject of the remainder of this chapter. For this comparison, we display only the results of ZOD and ZOL, as their test statistics have produced the best results to date. Full quantification of the performance differences between SAEs and optimization of their test statistics is an item of future work.

5.5 Results: Anomaly Detection

We applied our test statistics of ZOL and ZOD, as well as mean and median, to two different MAWI datasets. These datasets include two types DoS attacks at two different volumes. The *type* of attack is not critical; however, the volumes were specifically chosen to demonstrate the ability of our system to detect both low and high volume attacks. We first review our proof-of-concept methodology, then present the detection results and their analysis.

5.5.1 Proof-of-Concept Methodology

This proof-of-concept was designed to provide an end-to-end demonstration of some of the accuracy gains available from designing a network anomaly detection system that harnesses non-Gaussian methods. The following processing and measurement steps were used to generate our results, which are summarized in Figure 5.3:

1. Identify candidate MAWI traces, including attack and benign portions.
2. Download designated traces in .dump or .pcap format and truncate into benign and attack portions using *editcap.exe*, a command-line function of Wireshark.
3. Use a Python script to subdivide the attack and benign portions into N samples with M elements, counting the packets transiting the MAWI link per Δ_{sw}. The outputs of this script are a series of .txt files.
4. Ingest and process the benign and attack cases into a Monte Carlo routine per Section 2.5.2.
5. Process the Monte Carlo results using built-in MATLAB routines and generate the ROCs shown in this chapter.

The following technical notes may be of concern to the interested reader. All results in this section were generated using a Monte Carlo trial size $T = 20,000$ unless otherwise

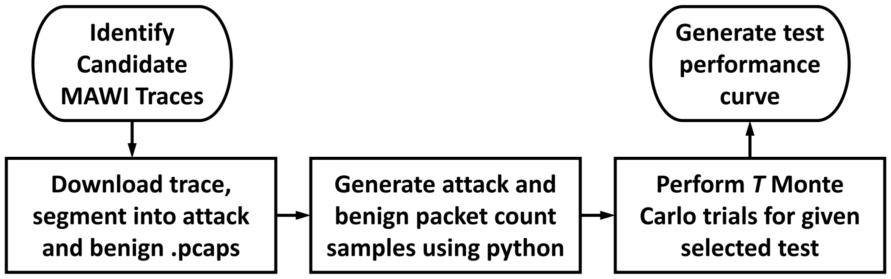

Identify Candidate MAWI Traces
Download trace, segment into attack and benign .pcaps
Generate attack and benign packet count samples using python
Perform *T* Monte Carlo trials for given selected test
Generate test performance curve

Figure 5.3. Major process steps in the implemented end-to-end detection system proof-of-concept

noted. Additionally, data was fit and generated during Monte Carlo analysis using the commercial program STABLE vice MATLAB; although, fits displayed in figures in this work were generated using the MATLAB MLE algorithms for convenience [13], [39]. Finally, all Python and MATLAB scripts that are referenced in this work are included in the Appendices.

5.5.2 Low Volume Anomaly: MAWI 20160428 Trace

The MAWI 20160428 trace was selected as our low volume attack case because the magnitude of the attack flow against the target IP address 163.32.146.254 varies between 15 and 25 % of the baseline traffic. Typical web servers run between 20% and 65% of capacity, with the lower figure being more typical of on-premises versus cloud servers [72]; thus, we consider a 20% volumetric DoS attack to be a reasonable representation of the approximate lower bound of an effective brute-force attack.

Approximately four minutes of the overall packet rate, in packets per 100 ms, is shown in black, with the isolated attack-only volume in red, in Figure 5.4. The attack commences at 05:08:32 and continues until the end of the trace. (Note: Unless otherwise stated, stated times are as given in the trace). This duration produced a large number of attack and benign reference cases, which we coarsely identified by labeling all cases prior to 05:08:30 as benign and all cases after 05:08:35 as attack.

Processing the benign (30 – 510 s) and attack (540 – 900 s) portions of the MAWI 20160428 trace using our chosen overlap and window parameters generated 763 attack and 1030 benign cases. For both the low volume and high volume scenarios, the settings used to process the

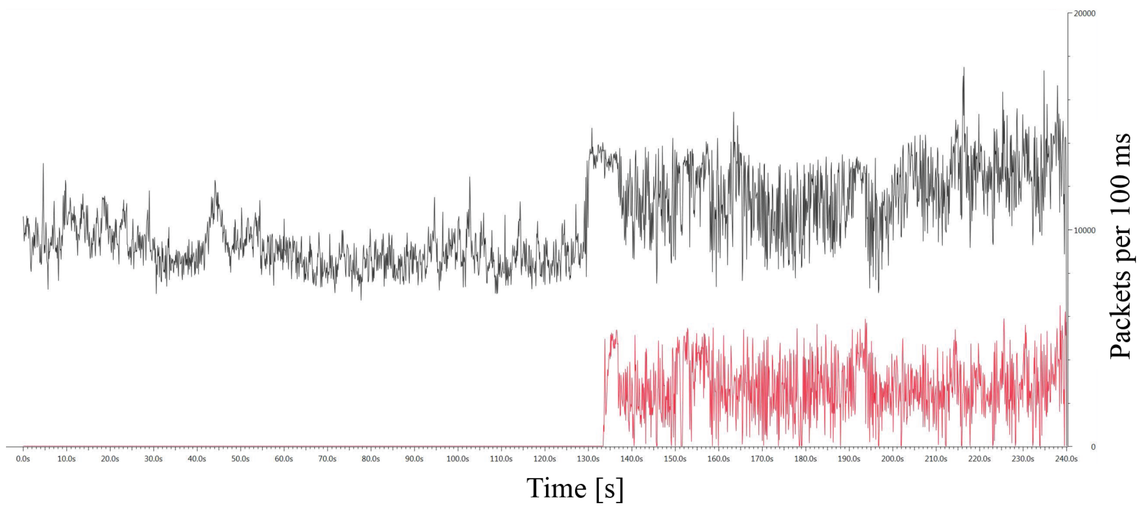


Figure 5.4. Change in packet rate due to low-volume DoS attack

traces as well as the resultant number of attack and benign samples are shown in Table 5.2. In this table, as in the entirety of this chapter, ϖ and Δ_{sw} are in units of seconds and milliseconds, respectively.

Table 5.2. Summary of analysis parameters and number of cases for examined scenarios

MAWI Trace	ϖ	Δ_{sw}	Δ_o	Attack Cases	Benign Cases
20160428	3	4	5/6	767	1031
20151114	3	4	11/12	1281	895

Using these benign and attack cases, we generated ROCs to compare the performance of the α-stable test statistics, ZOL and ZOD, to their equivalent classical test statistics of mean and variance. Figure 5.5 contains the only ROC showing the variance performance curve, as the results from this statistic were consistently inferior to the other alternatives. For this and all other ROCs, the performance of α-stable estimators is shown using solid lines, and estimator family (e.g., ZOD and variance) are grouped by color.

The results of Figure 5.5 are somewhat encouraging, particularly given the significant improvement in performance obtained by using the stable-derived ZOD instead of the variance. The results did not align with overall expectations for improvement, however, so investigation of concerns that arose during the Monte Carlo methodology development was required. This investigation identified an adverse effect that we call probabilistic

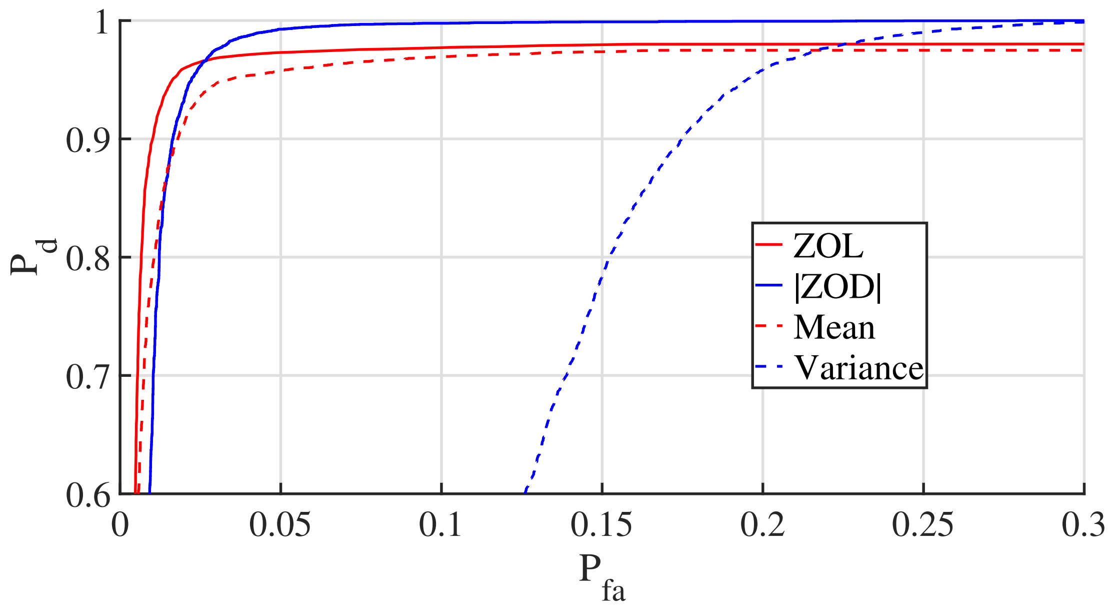


Figure 5.5. MAWI 20160428 (low volume) ROC comparison of stable and classical estimators when using generated samples

sample smoothing (PSS), the result of a *preliminary* Monte Carlo implementation that used randomly-selected STABLE fits to probabilistically re-create data samples. These *generated* samples, vice the *actual* samples that had been collected from the MAWI .pcap, were then used to generate the ROC in Figure 5.5.

5.5.3 Impact of Sample Smoothing

PSS results from this method of using the MLE fits of previously-collected data to generate new random samples for analysis. While generating data samples introduces additional randomness to the performance assessment and permits more trials (and conceivably a more-robust result), the generated-data method also has a tendency to remove *actual* outliers that could significantly affect the detection results. For this reason, we consider the generated-sample Monte Carlo approach to be a less realistic indicator of system performance. The generated-sample approach may, however, serve as an effective lower bound on performance for potential designs.

The effects of PSS are illustrated in Figure 5.6, which compares histograms of a randomly-selected actual sample and its generated equivalent, which was produced using the MLE

α-stable parameters of the actual sample. The generated-sample histogram and fits are given in black, while the sampled histogram and fits are shown in red. In this figure, the unintended effects of data generation on the dispersion and mode density are evident. Although the overall shifts are small, the effect on estimator results and detector performance were believed to be significant. To confirm this speculation, we altered the Monte Carlo algorithm to utilize only randomly-selected actual samples from the attack and benign pools per the method described in Chapter 2 and re-examined performance.

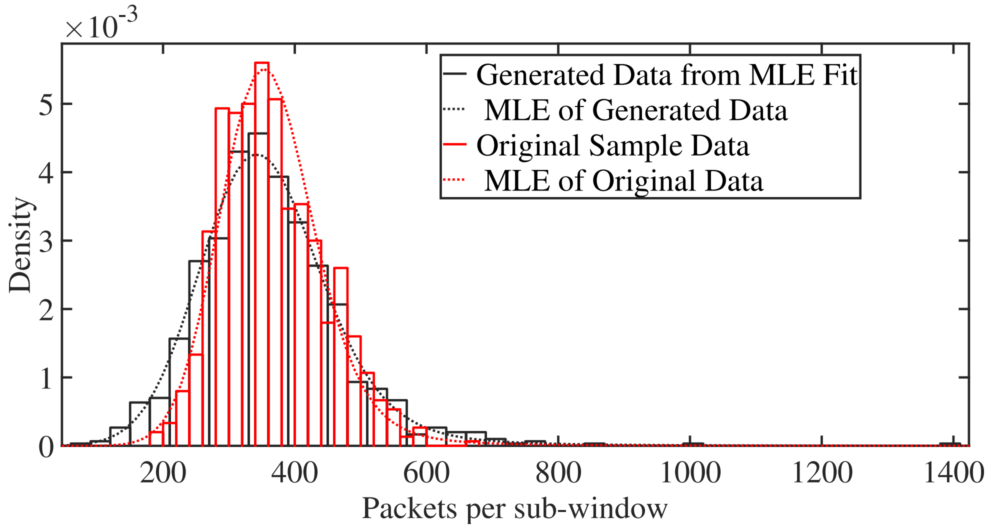


Figure 5.6. Comparison of original and simulated benign data samples

5.5.4 Monte Carlo Results using Actual Samples

The improved ROC from using the revised algorithm (i.e., actual vice generated data samples) for the low-volume attack scenario is shown in Figure 5.7, this time for a Monte Carlo implementation of 50,000 trials. Again, mean and ZOL curves in this figure are denoted by red lines, while dispersion estimators are shown in blue. The performance curve resulting from a test statistic based on the median is shown in grey (for display purposes) in Figure 5.7; the performance improvement of the α-stable estimators over this alternative robust statistic is notable.

ZOL Performance and Analysis in Low Volume Scenario

Two aspects of Figure 5.7 warrant further discussion. First, in this scenario, ZOL outperforms the mean and median over the entire performance curve. This is particularly

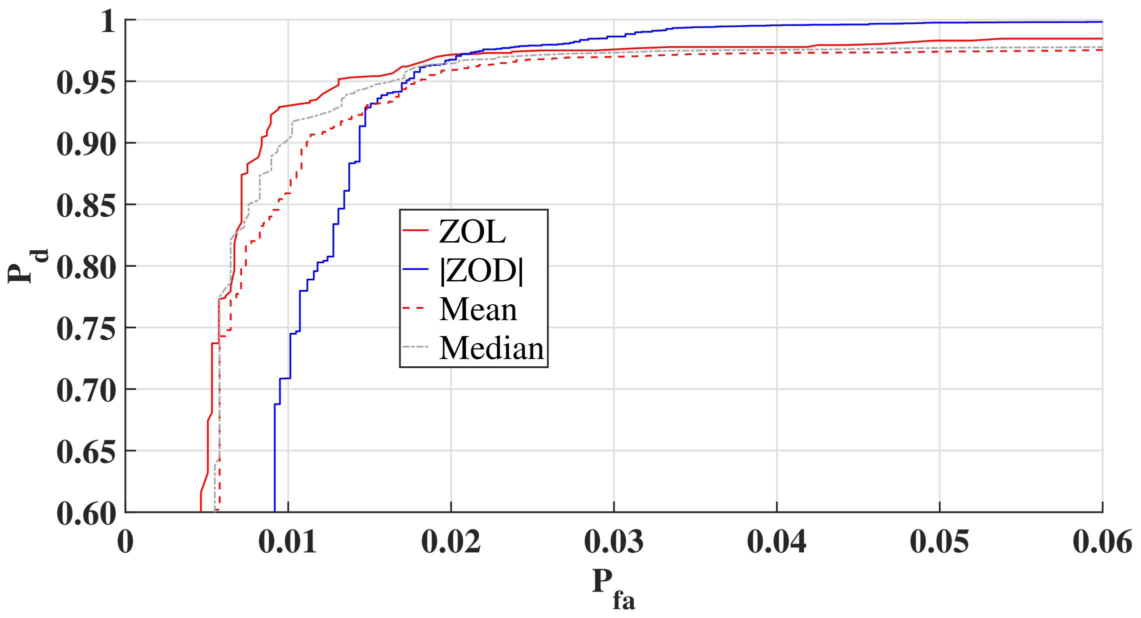


Figure 5.7. MAWI 20160428 (low volume) ROC comparison of stable and classical estimators when using actual samples

interesting because the mean is known to be the optimum location parameter for Gaussian distributions, and we established in Chapter 3 that attack traffic takes a Gaussian (or near-Gaussian) shape. For the scenario in Figure 5.7, nearly half (308 of 767) attack cases are near-Gaussian with α values > 1.95 (where the MLE fit begins to become less reliable based on empirical observation and the literature [13]). Effectively, even in a very-near Gaussian environment, ZOL performs at least as well as the optimal metric for a large number of the cases. This is a strong empiric demonstration of the effectiveness of ZOL and also demonstrates the literature consensus that traditional linear-based detection performance declines rapidly in the presence of non-Gaussian signals [21], [43].

Second, ZOL outperforms the median, though not by the same amount as the mean. The median is a frequently-used “robust statistic” and is optimal when the sample follows a Laplace distribution [23]. The performance of the SAEs relative to the median bear examination using additional datasets and samples, but it is clear from Figure 5.7 as well as our other empirical results that ZOL is a better robust statistic for our examined scenarios. (Note that these results regarding the median are repeated, though not shown, in the high-volume scenario that is discussed in the next section.)

ZOD Performance in Low Volume Attack Scenario

The performance of the |ZOD| test statistic (shown in blue) for the case in Figure 5.7 bears additional analysis on several points. First, both ZOL and |ZOD| are highly accurate. Their performance is comparable to or better than the best recent statistical AD results in the literature, as summarized in Table 5.3. The results from this research are summarized in the top four lines of this table.

Note that the peer studies in Table 5.3 were chosen based on publication quality, age, and rough comparability in terms of datasets and methodology. All approaches in this table are statistical except the last entry (i.e., NFBoost), which was a state-of-the-practice machine learning implementation using neural networks at the time of this dissertation [7]. The comparable results of Simmross-Wattenberg et al. [10] are given by the 2011 entries, with the volumes of the closest comparable DoS attack scenarios given by percentage in the "Algorithm" column. The results of the other comparable non-Gaussian statistical network anomaly detection implementation were conveyed in Figure 3.1 [5].

Table 5.3. Performance comparison of dissertation results to selected recent network AD studies

Study date	Detection Technique	P_d	P_{fa}	Algorithm	Source
2018	MAWI 160428	95.2	1.31	ZOL	-
2018	MAWI 160428	98.64	3.04	|ZOD|	-
2018	MAWI 151114	99.88	0	ZOL	-
2018	MAWI 151114	99.97	0.06	|ZOD|	-
2017	MAWI	79	20	Sketch Entropy	[73]
2014	KDD Cup	98.56	0.42 - 3.72	Entropy	[74]
2016	Proprietary	91	10	Multifractal	[75]
2016	Proprietary	89.6	12.6	Kalman Filtering	[75]
2016	Proprietary	88	16.7	NCAS	[75]
2017	Proprietary	78	0.55	Sketch&Bloom	[76]
2011	Proprietary	35	10	α-stable (25%)	[10]
2011	Proprietary	81	5	α-stable (100%)	[10]
2012	Mixed Public	98.8	1.9	NFBoost	[7]

A second observation that should be drawn from Figure 5.7 is that ZOD achieves perfect detection (as does variance, although at a higher P_{fa}). This implies that ZOD is an excellent discriminator in a multi-layer or ensemble detection system. Full examination of ZOD performance and development of multi-layer detection systems are items of future work, but these preliminary results are promising.

Third, the ZOD test statistic in Figure 5.7 is annotated as |ZOD| because it is a slight adaptation of the formula in (4.33). Traffic modeling research (described in Chapter 3) identified that attacks increase dispersion. The differencing method was shown to enhance the separation of attack outliers; however, while developing the proof-of-concept detection system, analysis of false positive (FP) and false negative (FN) cases identified that ZOD was not performing as expected. This discrepancy appeared to be due to the formula of (4.33) allowing positive and negative differences to offset when summed. ZOD was thus adapted to fully capture the dispersion changes during an attack by quantifying the *total magnitude* of change. |ZOD| is given by

$$\hat{\delta}_0(Z_t) = \hat{\phi}_{0,t}\left(\sum_{i=1}^{N}\left|Z_d - \hat{\lambda}_{0,t}\right|\right) \tag{5.13}$$

where

$$Z_d \sim \left(\boldsymbol{d}_t - \boldsymbol{d}_{(ref-1)}\right), \tag{5.14}$$

$\boldsymbol{d}_{(ref-1)}$ is the sample prior to the benign reference sample, and ZOP and ZOL are estimated from the sample under test.

Overall, ZOL and |ZOD| outperform their classical counterparts in the low volume attack scenario and imply potential for layering in a future ensemble detector. It is, however, appropriate to validate this assessment using other datasets and attacks. For the second scenario, we chose a high-volume DoS attack found in the MAWI 20151114 trace.

5.5.5 High Volume Anomaly: MAWI 20151114 Trace

An overview of our selected high volume attack scenario is shown in Figure 5.8, a plot of the packet rate per 100 ms over time for part of the MAWI 20151114 trace. A flooding attack begins 227 s into the trace and persists through the remainder of the trace. The benign period is characterized by a packet volume fluctuating around 185,000 packets per second, and the rate jumps 227 s after the trace start to approximately 370,000 packets per second. This attack is a periodic, volumetric attack with a typical rate of approximately 100% of the benign reference portion. The ON/OFF periods are approximately 27 and 5 s, respectively. While not shown due to space constraints, no portion of this trace after 21:03:47, other than OFF periods, has a rate less than or equal to the benign portion.

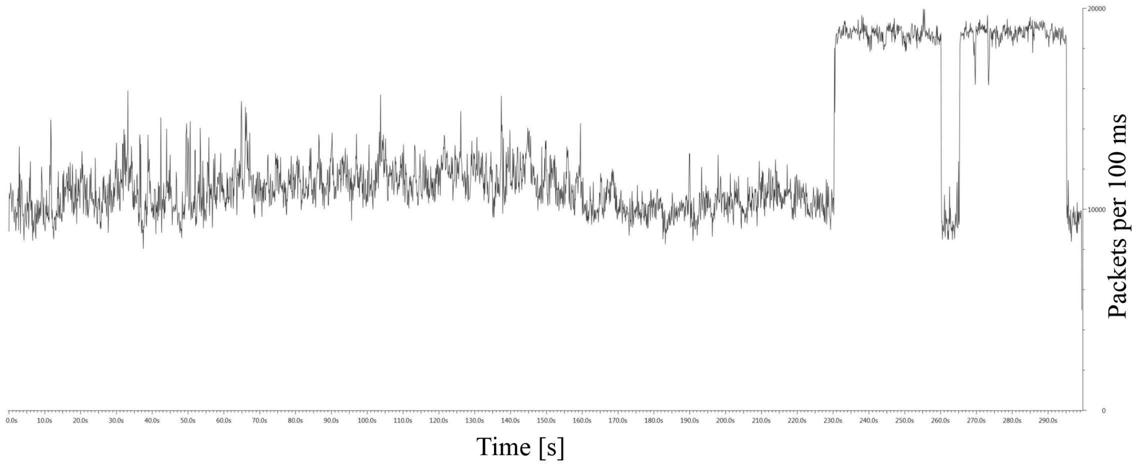


Figure 5.8. Volumetric overview of the benign and attack sections in the high volume attack scenario

A close-up volumetric display of the attack is shown in Figure 5.9, a plot of packets per second over time for the start of the attack portions of the MAWI 20151114 trace. This plot is for the time period from 21:04:00–21:08:00. The typical packet rate doubles during the attack, with ON/OFF periods of approximately 27 and 5 s continuing. The dark bars in the plot are counts of TCP errors automatically identified by Wireshark.

To develop our ROC for the high volume case, we divided the trace into attack and benign portions by time and subsequent rate analysis. All samples with an end time prior to 21:03:44 were labeled as benign; for the chosen ϖ and Δ_o, this time period produced 895 benign samples. We sampled the next six minutes of the trace for attack data; this period provided at least as many attack samples after purging invalid samples. Invalid attack samples (obtained during an OFF period) were removed after classification as such based on timing and measured ZOL, though these samples possessed additional indicators (e.g., significant drop in α, increasing mean-ZOL divergence, and bi-modal distribution) that may be used to build improved detection systems in future work.

The population of attack and benign cases used in our Monte Carlo analysis of test statistic effectiveness in the high volume scenario is given in Table 5.2. The results of the Monte

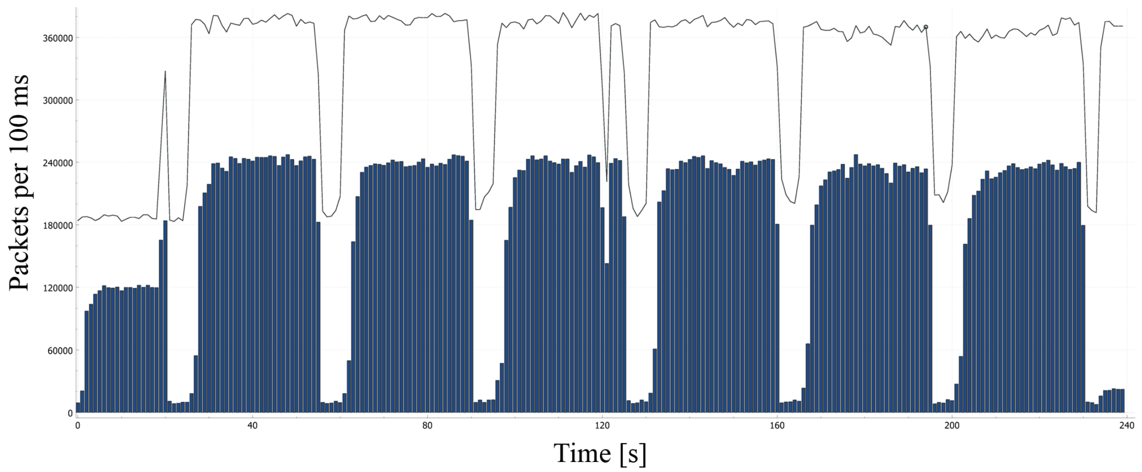


Figure 5.9. Detailed view of packet rate for portions of the periodic high volume DoS attack

Carlo analysis of the high volume trace are shown in Figure 5.10, illustrating that the classical detectors of mean and median provide slightly better performance than ZOL for larger P_{fa}.

Additional analysis of these results shows that, in this scenario, the high volume attack frequently produces a bi-modal distribution of packet counts per sub-window for the chosen sampling settings. In the presence of a bi-modal data distribution with the primary mode on the left (at a lower value), the ZOL and median robustness against outliers leads to a performance disadvantage, as the mean responds more dramatically to outliers. This effect is visually demonstrated in Figure 5.11, which is a plot of the packet volume histogram and location estimates from an attack transition (i.e., OFF-ON transition) sample of the MAWI 20151114 trace.

In this figure, the ZOL returns an estimate closely corresponding to the primary mode of the sample, while the mean estimate nearly splits the difference between the primary and sample modes. This window was labeled as an attack for our Monte Carlo analysis; extrapolating this case to the many transition periods included in this ON-OFF type attack explains the overall reduced ZOL performance relative to the mean for this scenario.

Returning to Figure 5.10, we see that |ZOD| performs nearly perfectly (i.e., P_d = 99.9% for P_{fa} = 0.001%). Recalling that |ZOD| is proportional to the magnitude of the difference

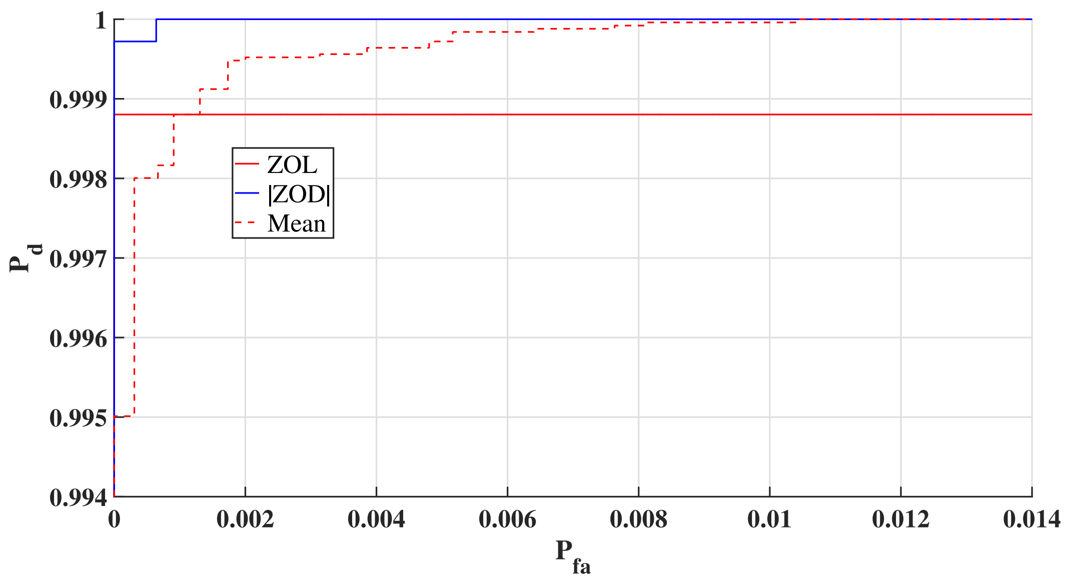


Figure 5.10. System ROC for the MAWI 20151114 high volume DoS attack scenario

between the outlying samples and the location estimate per (5.13), we can understand how the combination of a resistant ZOL and the large secondary mode in Figure 5.11 results in a very large value for |ZOD|. Put another way, |ZOD| requires the robustness of ZOL (and associated lessened performance in this scenario) to produce its superior result.

Overall, we have shown that the selected SAEs, when applied using suitable test statistics, outperform classic Gaussian estimators (e.g., mean and variance) as well as the robust classical estimator, the median. These performance gains were due to applying α-stable - based tests in our detection systems that were aligned with and adapted (slightly) to take advantage of the non-Gaussianity of the data. One important consideration of this result is that it can be extended to any discipline that performs similar processing of non-Gaussian, α-stable -distributed data.

Now that we have achieved one of our goals of a more accurate detection system, let us examine its scalability and potential for real-time application.

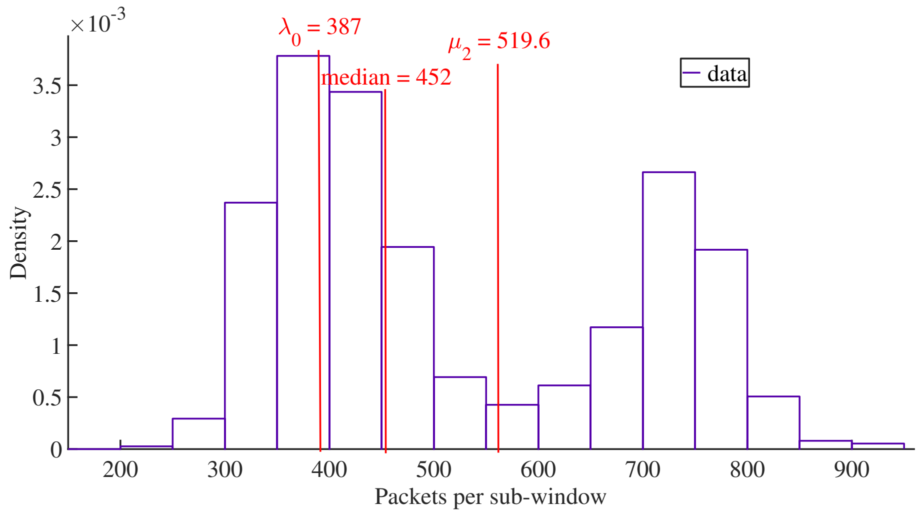


Figure 5.11. Comparison of estimator performance under bi-modal conditions frequently observed during the high volume attack scenario

5.6 Results: Real-Time Potential

The scalability analysis in this section considers only the times required to ingest and process input data files consisting of M counts of packets per sub-window, as our goal is to achieve a total processing time t_{tot} that is significantly less than the window size ϖ. As such, our t_{tot} should include windowing t_{win}, sample ingestion t_{imp}, sample estimation t_{est}, and testing t_{det}. The time required to actually measure and record features on the link and produce a .pcap of the traffic is highly dependent upon end-user hardware and implementation choices. Given our approach can process the input in a timely manner, the problem of obtaining hardware to support actual implementation requirements is delegated to the interested user.

Given these assumptions, the processes of sample ingestion, estimation, and detection were integrated into a MATLAB timing script (contained in the Appendix) that was used to measure t_p over 10,000 sequential runs and determine the average detection time $\bar{t}_d$ and average total processing time $\bar{t}_{tot}$ (which includes the *windowing* cost t_{win} of ingesting .pcap data and producing the feature vector). The results of this assessment are contained in Table 5.4, while the the average estimation times for each SAE developed in Chapter 4 (also averaged over 10,000 trials) are contained in Table 5.5. To identify any dependency of the computational cost on feature vector size (i.e., size of the sample in sub-windows), timing results are given for two values of M commonly used during analysis of the MAWI traces.

Table 5.4. Computational cost in μs of major detection process steps for a given sample size

M	ϖ	t_{win}	t_{imp}	t_{est}	t_{det}	$\bar{t}_d$	$\bar{t}_{tot}$
750	3	690,690	713	715	1	1,429	692,118
1,200	6	875,862	753	717	1	1,471	877,333

Table 5.5. Computational cost in μs of estimators when processing a given sample size

M	Mean	Myriad	LLE	ZOL	|ZOD|	ZOP	Stable Fit
750	5	9,300	379	725	715	5	42,055
1,200	5	-	290	709	717	4	-

As we consider the times in Table 5.4 and Table 5.5, it is important to consider that the scripts for .pcap processing as well as the LLE and all ZOS estimators were coded by the authors and are, as such, un-optimized. This implies that these timing results should be used for performance comparison only between order-of-magnitude groupings. Still, several conclusions can be drawn from Table 5.5.

First, any of these processes, with the possible exception of fitting an α-stable distribution, should be easily scalable to a real-time environment. Note that t_{win} was not determined from 10,000 trials due to performance issues associated with un-optimized code that resulted in significant slowdowns as .pcap size increased. Offsetting this issue, t_{win} includes the cost to open and read the .pcap file as well as the cost to write the output file to disk; a streamlined detection implementation would use smaller records and avoid the expensive file reads and writes. Because .pcap import was the costliest process, accounting for nearly half of the total counting time, we remain confident in our assessment that our methodology supports real-time implementation because even our un-optimized system can complete an end-to-end detection process on 3 and 6 s records in less than a second.

Some additional conclusions can be drawn from an examination of the total processing time for six estimators shown in Table 5.5; these results demonstrate that any of the given estimators can process a sample in less than one second using a personal computer. Note that myriad and stable fit times were not recorded for the 1200-element sample due to their large computational cost.

The second conclusion is that estimators can be grouped by performance based on cost increases of greater than an order of magnitude:

- The cheapest estimates take only a few microseconds and are obtained directly from the data: mean and ZOP as well as median and mode (not shown for display purposes).
- Single-parameter ML estimators cost a few hundred microseconds: myriad, LLE, ZOL, and by extension, ZOD.
- The most expensive process is the stable MLE fit, which costs a few thousand microseconds (approximately 500 times more expensive than the least expensive options).

Finally, ZOD is costly because it relies on the determination of ZOL as well as ZOP. While the larger execution time reduces its attractiveness slightly, ZOD is still implementable as a real-time estimator. Given the relatively modest performance specifications of the computer used to produce the analysis in this work, the results of this section demonstrate the scalability of our stable-based estimation and detection methods.

The interested reader should note that the results in this work were obtained on a personal computer running Windows 7 with 32 GB of installed memory and a 2.6 GHz Intel® Xeon™(E5-2640V3) 8-core processor. The Python version 2.6 and MATLAB r2017a codes used in this research were not optimized or coded for parallel computing. The use of commodity hardware and un-optimized code implies that the methods proposed in this research could be scaled to link rates much greater than 1 Gbps at substantially-reduced computational cost.

5.7 Summary

In this chapter, we developed our modular AD system model, which we then used to specify our data processing approaches. This allowed defining our input signal as well as test statistics, which were adapted to both an α-stable signal and our requirement for real-time feasibility. The performance curves of Section 5.5 demonstrated that stable-based estimators provide superior detection performance than classical approaches when processing the α-stable signals common in our selected network traffic traces. Finally, through a timing evaluation of our end-to-end proof-of-concept system, we showed that α-stable network traffic samples could be ingested, windowed, estimated, and tested in less than a second using an off-the-shelf personal computer.

THIS PAGE INTENTIONALLY LEFT BLANK

CHAPTER 6: Conclusions and Recommendations

This investigation began by exploring the ability of the α-stable distribution to accurately model network traffic across a range of aggregation periods, network traffic rates, and traffic types. The α-stable model was extended to include three public datasets (e.g., ISCX, MACCDC, and MAWI) and multiple traffic types, including benign, noisy, transition, and attack. This modeling work demonstrated that α-stable network traffic models are the most appropriate for the range of examined datasets and scenarios, as quantified using the LL ratio and average Hellinger distance.

The possibility of an α-stable parametric detection system was then assessed, using approximation or estimation of the α-stable parameters to reduce computational cost. Hypergeometric and Lévy decomposition methods for approximating α-stable distributions were examined; these approaches were abandoned after determination that their constraints and computational costs excessively restricted the flexibility of the intended implementation. Instead, sample estimators derived from non-Gaussian heavy-tailed distributions, namely the Cauchy and α-stable , were examined. A novel location estimator based on the Lévy distribution was also developed. The performance of these stable-derived estimators was evaluated with respect to traditional estimators such as mean and median in terms of computational cost as well as location accuracy.

The heavy-tailed estimators were used to develop and test a network anomaly detection system based solely on α-stable principles and non-parametric methods; i.e., no processes or algorithms were used in the detection system that relied on assumptions that samples or process errors followed a Gaussian distribution. A Monte Carlo evaluation was used to assess the end-to-end accuracy and computational cost of this system for two types and volumes of DoS attacks contained in real-world network traffic traces. The results of Monte Carlo evaluation were assessed using ROCs.

6.1 Contributions

Based on our review of the literature, this work provides a novel statistical, non-parametric α-stable network anomaly detection system. The sole previous α-stable implementation was a parametric system, evaluating the change in the ML α-stable fit and did not operate in real time [10].

6.1.1 Anomaly Detection

The proposed system is the first demonstration of a proof-of-concept non-parametric, non-Gaussian, statistical network anomaly detector with real-time capability. The estimators used, ZOL and |ZOD|, performed well in both non-Gaussian benign traffic and attack traffic (which tends toward Gaussian distributions). The *flexibility* of these estimators led to a significant improvement in statistical network anomaly detection accuracy as compared to the existing body of both Gaussian *and* non-Gaussian methods.

Results showed that the α-stable estimator of ZOL improved P_d by 6.5% in the low-volume scenario over that of the Gaussian-based mean, at a 1.0% P_{fa}. The α-stable |ZOD| estimator improved P_d by 10% in the low-volume scenario over that of variance at a fixed 2% P_{fa}. In the high volume scenario, |ZOD| had nearly-perfect performance; the remaining estimators demonstrated nearly equivalent performance, with ZOL outperforming mean for $P_{fa} \leq 0.1\%$

The proposed methodology also demonstrates the viability of real-time volumetric network anomaly detection based on α-stable principles. The implemented non-parametric system required a total, un-optimized processing time of less than 900 ms to import traffic captures (e.g., .pcaps), detect anomalies, and report results for blocks of 3- and 6-s network traffic samples collected from a 1-Gbps link. The un-optimized time to only handle and process feature vectors in the detection system was less than 2 ms.

In terms of the SAEs applied in the implemented system, this research contains a novel application of ZOS to network anomaly detection based on our review of the literature. The ZOL was found to improve detection performance by 3.8% and 0.25% over a comparable robust method, the median, for the low and high volume scenarios, respectively. Additionally, in this research we developed and demonstrated a novel, stable-based location estimator: LLE. From limited investigation, the LLE appears to provide accurate estimates of the location of the data mode and possess properties and performance similar to the other stable-derived sample myriad and ZOL estimators.

6.1.2 Traffic Modeling

The proposed methodology validates the improved network traffic modeling accuracy of α-stable distributions via examination of multiple cases of three publicly-available datasets for a range of normal and abnormal scenarios and aggregation periods using a quantitative performance comparison. The α-stable distribution was shown to accurately model network traffic at high-speed link rates (e.g., 1 Gbps) using analysis windows that are only a few seconds in length, two orders of magnitude smaller than windows used previously in stable-based work (and an order of magnitude smaller when accounting for differences in traffic rates).

As part of this α-stable modeling extension, it was identified that it is possible to improve the fit of the stable model by up to 50% through adjusting sub-window size, even as the traffic distribution tends towards Gaussian (i.e., during attacks). Preliminary investigation indicates that this same gain does not appear to apply to Gaussian models and definitely not to the same extent. Fit error of the α-stable model, as measured by d_{Hel}, was found to possess a global minimum in many of the examined traffic scenarios. This global minimum implies that adaptive methodologies can be applied to reduce fit error and improve detection system accuracy when employing α-stable parametric detection methods. Finally, limited investigation also demonstrated that α-stable model fit appears to be relatively insensitive to window size, justifying the use of smaller data windows to speed traffic analysis and, presumably, network anomaly detection without significantly sacrificing accuracy.

The exploration of traffic models also showed that, during an attack, the α-stable parameters α, β, γ, and μ change significantly and usually assume values that are significantly different than those of benign traffic. These trends imply that *combinations* of parameter shifts could be harnessed to develop a changepoint parametric anomaly detection system that advances the state of the practice.

6.1.3 Lévy Decomposition

In this research, we extended the scope of prior work in one other area besides α-stable traffic modeling. In this dissertation research, we considered an algorithmic implementation and practically demonstrated the Lévy decomposition of a PaS sample comprised of real data. The investigation of Lévy decomposition also included novel exploration of optimization techniques to reduce the computational cost and improve the accuracy of Lévy decomposition.

6.2 Recommendations

We remark that the majority of the results reported in this dissertation can be applied to *any* appropriately-distributed (i.e., α-stable) time series. Given the variety of areas in which the α-stable model has been successfully applied, the results of this research have the potential to affect many other fields of work, such as signal detection in wireless communications [18], [24] and rolling element fault monitoring in heavy machinery [77], given appropriate caveats.

6.2.1 Methodology Limitations and Mitigation

The proposed system must be considered in the context of its intended purpose: real-time, non-Gaussian detection of volumetric anomalies. The examined system detects traffic anomalies based *only* on changes in the volume or dispersion of one underlying feature, packets per sub-window.

Volumetric Detection and Data Aggregation

In its current implementation, the proposed system's effectiveness is limited, at best, against attacks that are designed to evade volumetric sensors, such as asymmetric workload attacks or expert attacks that rely on known weaknesses of the defended systems [78]. In principle, the methods of this research can be applied to detect other types of attacks using alternative features that respond to these types of attacks, such as bi-directional traffic volumes and response times. Alternative features, if properly selected, may even respond *only* to certain types of attacks, providing a combined classification and detection solution.

Related to this volume constraint, this implementation assumes an endpoint, aggregated view of traffic (i.e., that *all* packets destined for a target are visible to the system). Distributed attacks on a multi-homed target, or attacks contained in traffic collected from a monitoring point upstream of the defended target, are unlikely to be detected by this system. In the event of distributed or stealthy attacks, pre-detection data aggregation from distributed sensors may enable effective, alternative detection implementations that are still aligned with the *principles* of this research.

Conversely, disaggregation of data through methods such as hierarchical heavy hitters (HHH) could enable detecting small-relative-volume attacks that are normally not identifiable in aggregated traffic using the methods in this work [79]. Similar to HHH methods, disaggregation can also potentially be used in conjunction with algorithm optimization to scale the proposed system to full backbone speeds (i.e., terabits per second), as required [79].

"Optimal" Detection and Overfitting

In terms of the examined attacks, alternative detection approaches, such as entropy of ports and IP addresses, or signature-type detectors, may obtain better results against the specific attacks contained in the examined MAWI traces. The strength of the proposed methodology in its perceived implementation is that it may serve as a more accurate *alerting first layer* in a multi-layer (i.e., multi-detector) intrusion detection system. More generally, a statistical methodology has the further advantage of being un-reliant on a signature; i.e., the proposed statistical system can detect previously-unseen attacks with a volumetric component.

This statistical methodology may be extended to other detection approaches based on the expectation that many other features of network traffic can be modeled using α-stable distributions based on the heavy-tailed nature of the controlling process (e.g., TCP retransmission [2]); thus, the test statistics and general methodology of this research could be easily adapted to create additional detection layers (i.e., for attack classification) that more accurately respond to changes in heavy-tailed features.

As more features are added to a detection system, model overfitting and assessing the independence of these features become significant concerns [60]. We assess that the proposed single-attribute detection system is unlikely overfitted; the consistent results across both datasets (both in terms of absolute as well as *relative* performance) despite different volumes and types of attacks reinforce, but admittedly do not *prove*, this assertion. Overfitting remains a valid concern, as proposed extensions of this research include developing multi-attribute *ensemble* detection systems such as those proposed in Section 6.2.2.

Overfitting is a also potential issue with the modeling methodology in this dissertation. Given the background literature of α-stable modeling discussed in Chapter 1 ([10], [12], [16], [17], [21], [22]) as well as the results in Chapter 3, it is our conclusion that the Gaussian model underfits network traffic data by failing to account for the heavy tails and, frequently, skew. As such, the α-stable model is *more appropriate* than the Gaussian but still may possess the drawback of overfitting the data.

In sum, we acknowledge the *potential* for the 4-parameter α-stable model to overfit network traffic data while deferring detailed overfit analysis to future work for the reasons discussed above. Evaluation of overfit goes hand-in-hand with the extension of the proof-of-concept system in this dissertation to additional datasets, attacks, and detection systems and is one of several items of future work.

6.2.2 Future Work

Two areas of this investigation were not as productive as we had hoped. First, we were unable to develop a PaS approximation method for the $\alpha \in (0.5, 1.0)$ case during the allotted investigation time. The α-stable parameterization issues (see Chapter 2), combined with complex-valued results induced by decomposing negative-valued mixing functions, prevented easy extension of the methods from the $\alpha \in (0, 0.5)$ case.

The second under-productive research area was that of adapting ZOP to network anomaly detection. ZOP produced perfect detection results in the low volume scenario but only at a higher P_{fa} than alternative SAEs. The limited time available for this research precluded full investigation into the causes and potential solutions. Given the importance of Gaussian-derived power in numerous real-world applications, a more detailed investigation of the issues surrounding our ZOP results seems to be warranted. Also, ZOP's lack of direct correlation to ZOL and |ZOD| performance suggest potential for application of this statistic in an ensemble system.

There are several other lines of investigation that were not fully explored during this research project due to time constraints; these topics have also been incorporated into our recommendations for future work.

PaS Decomposition

In our examined datasets, samples with $\alpha \in (0.5, 1)$ were more typical than $\in (0, 0.5)$. Also, very heavy-tailed datasets are common in the field of natural processes, and the performance of Gaussian methods increasingly declines as $\alpha \rightarrow 0$ [43]. Development and evaluation of a PaS decomposition algorithm for $\alpha \in (0.5, 1)$, as well as an examination of the accuracy impacts of extending this methodology to approximating α-stable distributions with characteristic exponent $\alpha > 1$, could have broad applicability both in and out of the networking field. Part of this investigation could include approximation optimization and quantification of the maximum achievable accuracy of this methodology. Ultimately, this closed-form approach could enable fast, accurate probabilistic applications such as detection in Lévy environments and Bayesian analysis.

Non-Parametric α-stable Estimation

A detailed functional exploration of the characteristics and performance of the various SAEs examined in this dissertation would enable determination and application of the "best" SAE for a given scenario or dataset. It bears repeating the problem and methods of α-stable detection apply to *numerous* fields besides networking, particularly when real-time implementations are supported.

Future modeling and estimation applications could be supported by, first, a full performance and accuracy exploration of the LLE and alternatives, particularly for data characterized by small α. It is important to fully quantify the performance *differences* between SAEs, as well as the relative impacts of differencing. This investigation would be aided by full dissection of the FP and FN cases from the MAWI 20160428 and 20151114 datasets, an investigation that was not completed during the course of this research due to time constraints. Resolving these FPs and FNs would be the first part of a full performance exploration of, and optimization of, SAE test statistics, including ZOP, using additional DoS and non-DoS attacks and datasets.

Improved α-stable Network Anomaly Detection Methods

The performance demonstrated by the implementation in Chapter 5 remains inadequate in terms of the ever-increasing volumes of network traffic and threats. A multi-layer detection system using SAEs may improve overall system accuracy or at least improve P_{fa} at the current P_d. Parametric α-stable detection may provide another method of improving performance. Additional examination of the changes in α, β, and γ using a broader range of scenarios and datasets may identify methods to distinguish between traffic scenarios (e.g., transition, benign, and attack). It would ultimately be interesting to compare the maximum achievable performance of the parametric and non-parametric implementations in terms of speed, P_{fa}, and P_d.

Implementation and assessment of an automatically-adaptive α-stable detection system is another interesting topic that could improve α-stable detection performance. Empirical analysis over the course of this project (even if not discussed in this dissertation) showed that adaptive windows, sub-windows, and memory have the potential to affect system accuracy. Quantification of the available gains, and application of these methods to additional datasets and types of attacks, is a relatively narrow research topic that should yield novel methods and further accuracy gains.

One final topic that has the potential to contribute to network anomaly detection theory and accuracy is an investigation of the independence (i.e., orthogonality) of different α-stable indications of the same network anomaly. Feature independence does not seem to have been treated rigorously in the literature [80]–[82], yet it has significant implications for all ensemble approaches.

APPENDIX: Modular Detection System and Code for Lévy Mixture Approximation and Proof-of-Concept Detection System

THIS PAGE INTENTIONALLY LEFT BLANK

Modular Detection System and Supporting Code

In this appendix, we provide details supporting the detection system, such as a more detailed detection system diagram and the code used to support the end-to-end detection system proof-of-concept. This code is divided by function. The implementation for α-stable estimation discussed in Chapter 4 is in Section A.2. Finally, the Monte Carlo probabilistic analysis code used to obtain the detection results in Chapter 5 is contained in Section A.3.

A.1 Modular Detection System

The modular detection system in Figure A.1 is a more detailed version of the detection system shown in Figure 5.1. This modular system is intended to serve as an implementation framework for adaptive features and additional layers discussed in Chapter 5 and Chapter 6, items of future work identified in this dissertation.

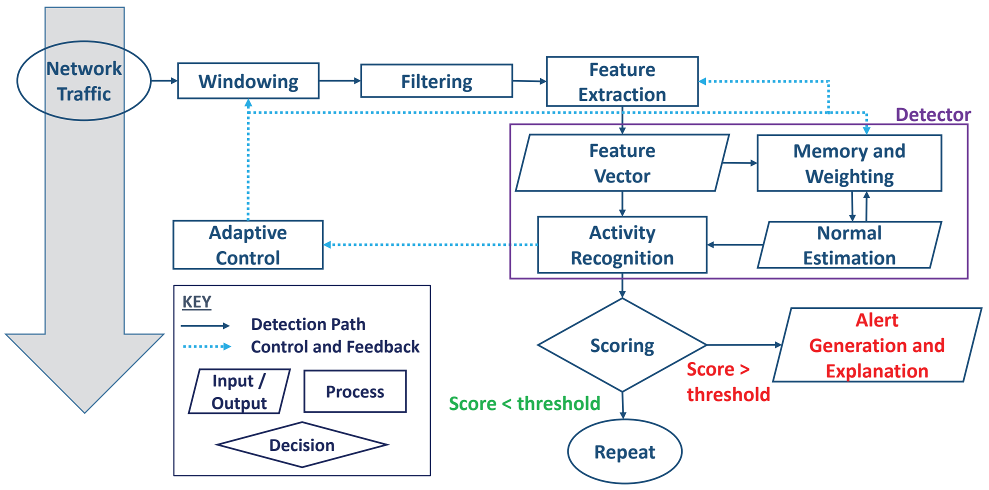


Figure A.1. Modular network anomaly detection system model developed in this research. Source: [20].

A.2 MATLAB Code for LMA

The LMA algorithm code is contained in this subsection. It is only valid for $\alpha \in (0, 0.5)$, as discussed in Chapter 4 and Chapter 6.

```
%% This script decomposes a file with data distributed at
    alpha < 0.5 and develops a Positive alpha-stable
    decomposition using weighted sums of Levy distributions
    per [Kuruoglu 2003]

clear;
% Define new variable N, = number of sampling points
N = 10;
  %% Our sampling interval will be from 0 to 10
  %y_i = linspace(min(axxis),max(axxis),N); [27 Mar]

right_I = 3; % Assigns end of interval
left_I = 0.1; % Assigns start of sampling interval

NumPoints = 500; % Number of points on axxis where we
    calculate PDFs
lastPoint = 20;  % Last Point of axxis that we are fitting

%% Generate fY(y) Stable PDF for uniform sampling
% Also generate fX(x) for comparison, as well as fZ(z).
    Requires:
%
% * Parameter table

%% Define Distributions and RVs
% Fixed Params %
% Set Parameters for Stable RVs X (Levy), Y (PaS), Z (PaS)
% a = alpha, b = beta, d = dispersion, s = shift
% Location is calculated/adjusted for k = 1 parameterization
     per
% Nolan[2002]
%              a   , b, g, mu
ParamTable = [0.5, 1, 1, 0   ; % RV X given as Levy
```

```
                0    , 1, 1, 0   ; % RV Y, PaS, aY from aZ
                0.35, 1, 1, 0 ]; % RV Z, PaS, aZ set to 0.4
ParamTable2 = [0.5, 1, 1, 0   ; % RV X given as Levy
                 0    , 1, 1, 0   ; % RV Y, PaS, aY from aZ
                 0.35, 1, 1, 0 ]; % RV Z, PaS, aZ set to 0.4
%%
% * Updated dispersions per [Kuruoglu, 2003]
% * Corrected displacements to convert from k=0 to k=1
   parameterization per
% Nolan

%% Compute corrections and derived constants

  % aY = aZ / aX per [Kuruoglu, 2003], originally [Hardin,
     1984]
ParamTable(2,1) = ParamTable(3,1)/ParamTable(1,1);
ParamTable2(2,1) = ParamTable2(3,1)/ParamTable2(1,1);

  % dY given by [Kuruoglu, 2003] (4) if aZ < 0.5 or (11) if
     0.5 < aZ < 1
if ParamTable(3,1) < 0.5
   dY = (ParamTable(3,3) * cos(ParamTable(3,1)*pi))/((2^
       ParamTable(3,1))*cos(ParamTable(3,1)*pi/2))
elseif ParamTable(3,1) < 1.0
        errordlg('We␣have␣not␣yet␣introduced␣functionality␣
           for␣1␣<␣alpha(Y)␣<␣2.');
else
        errordlg('Invalid␣alpha(Z)');
end

  % Update ParamTable for calculated dY
ParamTable(2,3) = dY;
ParamTable2(2,3) = dY;
```

```
  % Now shift Distributions from k=0 -> k=1 by adjusting
     Location for all RVs in ParamTable
for i = 1:size(ParamTable,1)
    ParamTable(i,4)=ParamTable(i,4) + ParamTable(i,2)*
       ParamTable(i,3)*tan(ParamTable(i,1)*pi/2);
end
%% Now that we've parameterized the Stable Distributions
   which form the basis for our Mix, let's create the actual
    PDFs.
%%
%% Make PDFs
  % Create distribution table
for i = 1:size(ParamTable,1)
    DistTable(i,:)= makedist('Stable','alpha',ParamTable(i
       ,1),'beta',ParamTable(i,2),'gam',ParamTable(i,3),'
       delta',ParamTable(i,4));
end

for i = 1:size(ParamTable,1)
    DistTable2(i,:)= makedist('Stable','alpha',ParamTable2(i
       ,1),'beta',ParamTable2(i,2),'gam',ParamTable2(i,3),'
       delta',ParamTable2(i,4));
end

% Set distribution size
axxis = linspace(0,lastPoint,NumPoints);
axxis2 = linspace(-2,8,NumPoints);
% ...and finally, make PDFs
for i = 1:size(ParamTable,1)
    PDTable(i,:) = pdf(DistTable(i,:),axxis(1,:));
end

for i = 1:size(ParamTable,1)
```

```
    PDTable2(i,:) = pdf(DistTable2(i,:),axxis2(1,:));
end
%% And now, let's plot them.

figure
hold on
plot(axxis,PDTable(1,:),'k');
plot(axxis,PDTable(2,:),'m--');
plot(axxis,PDTable(3,:),'r-.');
%plot(samplePoints,v_i,'bx');
axis([0,6,0,1.2])
title({'Mixing␣Functions␣for␣Fig.␣2␣[Kuruoglu,␣2003]','(
   distributions␣shifted␣for␣k=1)'})
lgd.FontSize = 11;
legend('X:␣\alpha␣=␣0.5','Y:␣\alpha␣=␣0.8','Z:␣\alpha␣=␣0.4'
   ,'Location','northeast')
hold off

%% Plot the unshifted components for comparison

figure
hold on
plot(axxis2,PDTable2(1,:),'k');
plot(axxis2,PDTable2(2,:),'m--');
plot(axxis2,PDTable2(3,:),'r-.');
%plot(samplePoints,v_i,'bx');
axis([-2,6,0,1.2])
title({'Mixing␣Functions␣for␣Fig.␣2␣[Kuruoglu,␣2003]','(k=0␣
   (Unshifted)␣Parameterization)'})
lgd.FontSize = 11;
legend('X:␣\alpha␣=␣0.5','Y:␣\alpha␣=␣0.8','Z:␣\alpha␣=␣0.4'
   ,'Location','northeast')
hold off
```

```
%%
% Notice that there is something "unusual" going on here...
    because the shift
% correction for different parameterizations is $\( \beta\
    gamma\tan\Big(\frac{\alpha\pi}{2}\Big)
% \)$...the various Stable distributions don't shift by
    equal amounts.
%
%
%% Let's get the Mixing Ingredients!!
% First, we must sample fY(y) uniformly to obtain our
    components.
%
% *  Note 1: Because of the calculated shift fY(y) < 0.6 is
    zero, some mixing
% components may create zero/infinite values and will have
    to be discarded to
% make the code work.
% * Note 2: Because [Kuruoglu,2003] defines Stable
    distributions in the k=1
% Parameterization per [Nolan, 2002], we will perform our
    calculations based on
% the shifted Stable distributions contained in the matrix
    variables _ParamTable
% _and_ PDTable_ (in the code).
% * Note 2a: [27 Mar] To attempt to improve the fit,
    altering code to adjust
% # of Components (_N_) and the sampling interval (now
    called _I_).  [K03, K98,
% and B06] make no reference as to how to determine the
    sampling interval, just
% that fY(y) should be sampled uniformly.
%
```

```
% We will start by defining the Number of Mixture Components
      (_N)_ and then
% setting uniform sample points.  For now, we will define
    the uniform sampling
% interval to be the same as our initial axis values [0,10]
    since values > 6 are
% largely trivial.  These sampling points are actually our
    y_i values that have
% a role in generating our mixing constants.
%%
  % Define new variable N, = number of sampling points
%N = 40; % Assignment moved to top
  %% Our sampling interval will be from 0 to 10
  %y_i = linspace(min(axxis),max(axxis),N); [27 Mar]

%right_I = 20; % Assigns end of interval; ASSIGNMENT MOVED
    TO TOP
%left_I = 0; % Assigns start of sampling interval;
    ASSIGNMENT MOVED TO TOP
  % Accordingly, the sampling interval is now between the
      left and right eyes
y_i = linspace(left_I,right_I,N);
%%
% Now we need to actually sample our Stable fY(y)
    distribution to generate
% fY(y_i) values for i = 1 to N

% Sample mixing distribution = fY(y) at y_i uniformly-
    distributed points
  % Compute y_i values by sampling Y at calculated uniform
      points
fY_y_i = pdf(DistTable(2,:),y_i);
```

```
%% Plot the original components and the sample points

figure
hold on
plot(axxis,PDTable(1,:),'k');
plot(axxis,PDTable(2,:),'m--');
plot(axxis,PDTable(3,:),'r-.');
plot(y_i,fY_y_i,'b*');
%plot(samplePoints,v_i,'bx');
axis([0,6,0,1.2])
title({'Mixing␣Functions␣for␣Positive␣alpha-Stable␣
   Approximation','(distributions␣shifted␣for␣k=1)'})
lgd.FontSize = 11;
legend('X:␣\alpha␣=␣0.5','Y:␣\alpha␣=␣0.8','Z:␣\alpha␣=␣0.4'
   ,'Mixing␣Sample␣Points','Location','northeast')
hold off

%%
% We now have y_i values and their corresponding fY_y_i
   values.  We need
% to map these values to their corresponding v_i and fV_v_i
   values by applying
% the transformations in [Kuruoglu,2003] given by Eq.(6) and
    V = Y^2.
% First, v_i:

  % Now transform y_i to v_i from [Kuruoglu, 2003]
     relationship V = Y^2
v_i = y_i.^(2);
%%
% And now, fV(v_i):

 % Calculate fV(v_i) from [K03] Equation (6)
```

```
for i = 1:N
    fV_v_i(i) = fY_y_i(i)/(2*sqrt(v_i(i)));
end

K = sum(fV_v_i);
%%
% Now, I've made a couple of shortcuts via the code:
%
% * Note 3: replaced fY(sqrt(v_i)) in [K03] Eq. 6 with fY(
    y_i) values calculated
% earlier due to the transformation that sqrt(v_i) = y_i
% * Note 4: The use of _N _discrete y_i or v_i values
    creates discrete sample
% points, effectively, for fV(v_i).
% * Note 5: I calculated K by summing the fV_v_i values.  K
    is the normalization
% factor, and has been determined by adapting [Kuruoglu,
    1998] and [Boubchir and
% Fadili, 2006].
% * Note 6: If K = 1, then the Normalization Constant has no
      effect
%
% Ultimately, now, we have our mapping of y_i -> v_i -> fY(
    y_i) -> fV(v_i)
% values for _N_ mixture components:
%%
%% Finally, the Mixing!
% Let's start by defining the basis for what we are mixing,
    a Levy distribution.
% This is a closed form of the Stable(0.5,1) distribution fX
    (x).  The whole point
% of this work is to simplify calculations by using N mixing
      components of scaled
```

```
% Levy distributions.
%
% The Levy distribution is defined in [Kuruoglu, 2003] in Eq
    . (2), where
% dispersion _(d)_ = 1.  In the definition below, I've
    paramterized it to support
% arrays as well as varying values of dispersion (_d_) and
    mixing parameters _(v)_.
%%
Levy_dv = @(x,d,v) ((d/(2*pi))^0.5)*(1./(x./v).^1.5).*exp(-d
    ./(2*x./v));

% Currently we will fix d = 1
d = 1;
%%
% Now we'll calculate the fX(z/v_i) values for _N_ Levy
    distributions using
% our original plotting axis.

  % Calculate scaled fx(z/v) values for each row
for i = 1:N
    % Update v for next scaling factor
    v = v_i(i);
       % Calculate new scaled fx(z/v) for new scaling factor
    fx_scaled_X(i,:) = Levy_dv(axxis,d,v);
end

%% ### Note that multiple plots used to visualize the
    progress of the code have been removed.
%% ### Contact Chad Bollmann for the original code.

% Now that (removed) is promising... note that they have been
      shifted by the various
```

```
% v_i, but not yet _weighted_ by the corresponding terms
    from [Kuruoglu, 2003]
% Eq. 9, adapted to be $\frac{f_V(v_i)}{v_i$. Note that we
    previously calculated
% each fV()...we now need to simply divide each term by its
    appropriate v_i and
% then weight the Scaled Levys by their corresponding term.

scaleFactors = zeros(1,N);
for i = 1:N
    scaleFactors(i) = fV_v_i(i)/v_i(i);
end
% This calc was checked, as were all of the above with the
    exception of the Raw Levy generation, which was
    previously checked.

%%
% This seems promising, since the Scale Factors computed
    with different
% weights; additionally, it passes the gut check because the
     highest weight is
% left-skewed, consistent with positive stable distributions
    .
%%
% Analysis: These values (plot removed) seem reasonable,
    with values on the left of the
% plot having a larger magnitude than values that would be
    obtained from the tail
% of the sampled distribution, fY(y).
%% "So, there we were..."
% So now, let's put some things together and see where we
    are. We have the
```

```
% raw mixture components composed of Levy distributions for
    various scalings of
% v_i (_fx_scaled_X)_, the weighting scalars (_scaleFactors_
    ). Why don't we multiply
% them together and see what we get? Intuitively would
    expect:
%
% * Components to the left, such as Component 2, to have the
      highest weight
% and provide the most contributions. Whereas Components
    captured more from the
% tail to have lower weights and mainly shape the "tail" of
    the weighted distribution
%%
% First we have to eliminate any pesky NaN in scaleFactors
  % Nice recursive loop from Stephen Cobeldick via MathWorks
      , thanks
scaleFactors(isnan(scaleFactors)) = 0;
scaleFactors(isinf(scaleFactors)) = 0;
  % Presize our output matrix for speed
scaledZ_components = zeros(N,size(axxis,2));
  % And compute
for i = 1:N
    scaledZ_components(i,:) =  scaleFactors(i).*fx_scaled_X(
        i,:);
end
%%
% .
%% And, finally...
% Let's just straight-up sum the weighted Levys
% along the x points given by _axxis_ and see what the
    result is:
%%
```

```
  % Again, get rid of any pesky NaN and recompute K
scaledZ_components(isnan(scaledZ_components)) = 0;
fV_v_i(isinf(fV_v_i)) = 0;
K = sum(fV_v_i);  % Based on [Kuruoglu 1998] and [Boubchir
   and Fadili XXXX]
fZ_approx = sum(scaledZ_components);
  % Compute the normalized values
fZ_approx_normd = fZ_approx./(K);
  % And plot
figure
hold on
     plot(axxis, fZ_approx)
     plot(axxis, fZ_approx_normd)
     plot(axxis, PDTable(3,:),'k.')
axis([0,8,0,1.0])
%set(gca, 'YScale', 'log')
t1 = strcat('for␣N␣=␣',num2str(N),'␣Components');
%, Sampling Interval = [',num2str(left_I),',');
%t1 = strcat(t1, num2str(right_I),'], K = ',num2str(K));
title({'Sum␣of␣Weighted␣Levy␣Components␣versus␣Reference',t1
   });
lgd.FontSize = 11;
legend('Levy␣Sum','Normalized␣Sum','Reference␣fZ(z)~S
   (0.35,1,1,0.6128)','Location','northeast')
hold off
%% Notes on K:
% (a) K is correct as coded.  The accuracy of the
   approximation is tremendously
% affected by the location of sample points obtained from
   the mixing
% function.
% (b) Best locations are near the peak of the mixing
   function, then
```

```
% approximately balanced and alternating on each side
```

A.3 MATLAB Code for Monte Carlo Implementation

The code in this section takes, as an input, the .txt files from *MAWI_Reader*, applies SAEs and test statistics, then generates performance curves from the results. Two types of performance curves are produced: Curves derived from actual sample data, and curves from generated data. A discussion of the advantages and disadvantages of each approach is contained in Chapter 5.

Benign and attack samples must be stored in different folders, and the folder paths must be updated in order to use this script. Additionally, for the generated data implementation, this script was designed to ingest α-stable ML fits from an .xlsx file containing all of the attack or benign sample fits. The fit for each sample should be placed in a different column. This script assumes that the parameter order is $\alpha, \beta, \gamma, \mu$ for rows 2–5, respectively, of each column.

```
% This is the Monte Carlo code to (1) ingest .txt files of
    real data samples of network traffic windows counting
    pkts per sub-window, (2) apply test statistics, (3)
    develop ROCs to assess results (4) ingest .xlsx files to
    generate data samples and repeat above
% * Note that L must be set to the lengths of the ingested
    data files
%% 1. Create randomization and Monte Carlo control variables

clear;
% of trials
    N = 50000;
% Number of data points in each trial
% *** NOTE THAT THIS PARAMETER MUST BE VARIED BASED ON THE
    RECORD LENGTHS IN THE INGESTED DATA FILES  ***
% The low volume 160428 scenario files have 750, 1500, and
    1200 element files available
    L = 750;
% Shuffle the random numbers so they don't repeat
rng('shuffle')

%% 2. Real Life Data Case
%% A. Set up data array - Ingest the data
% Read data from folder containing .txt files with packet
   counts.  These files
% were created by Mark Kragh's python program.
%
% * Start with Benign Files (Script generated by MATLAB)
%%
% Read all the files in the specified folder:
%%

Folder = 'D:\MAWILab\20160428\3secSegs\BenignSegs\3s_0.5
   overlap_4msSubWindow\';
oldFolder = cd(Folder);
Files = dir('*.txt');
 % End result is a Nx1 structure with the .txt file names in
     the first
 % column. Working directory is stored in 'oldFolder'

 % Now we will create variables from all of the applicable
    files in the
 % directory.  And determine sub-window size from the values
     in the left
 % column
numFiles = length(Files);
BenData = zeros(10000,numFiles)-1;  % Create a placeholder
   array of negative numbers...max array size is 10000!!

  % Into our placeholder array of length 10000, let's insert
```

```
        the count from
   % each CSV file in our folder into array "fileLengths"
fileLengths = zeros(1,numFiles);
for k = 1:numFiles
    x = csvread(Files(k).name);
    fileLengths(k) = length(x); % Measure the record length
        and record
    BenData(1:fileLengths(k),k) = x; % Append new record
       into Input Matrix
end

%%
% So now we have all of the benign records in "Data". Let's
    truncate the
% inputs to the minimum length to ensure uniform record
   lengths.

smallest = min(fileLengths);
BenData=BenData(1:smallest,:);

% Now check each file to verify it is the same length
  % refSize = sum(Data(:,1)>=0); % Deprecated code to count
       number of
  % elements != -1 in matrix column

refSize = smallest;
difFileLengths = 0;
for i = 1:numFiles
    if size(BenData(:,i)) ~= refSize
        fprintf('***WARNING:␣THIS␣WARNING␣SHOULD␣NOT␣APPEAR.
           ␣NOT␣ALL␣INPUT␣FILES␣ARE␣OF␣SAME␣SIZE␣!!␣\n')
        difFileLengths = 1;
    end
```

```
end

%% Now, repeat for the attack data

Folder = 'D:\MAWILab\20160428\3secSegs\AttackSegs\20160428
   _Attack_3s_4ms\';
oldFolder = cd(Folder);
Files = dir('*.txt');
 % End result is a Nx1 structure with the .txt file names in
     the first
 % column. Working directory is stored in 'oldFolder'

 % Now we will create variables from all of the applicable
    files in the
 % directory. And determine sub-window size from the values
     in the left
 % column
numFiles = length(Files);
AtkData = zeros(10000,numFiles)-1;  % Create a placeholder
   array of negative numbers

  % Into our placeholder array of length 10000, let's insert
      the count from
  % each CSV file in our folder into array "fileLengths"
fileLengths = zeros(1,numFiles);
for k = 1:numFiles
    x = csvread(Files(k).name);
    fileLengths(k) = length(x);  % Measure the record length
        and record
    AtkData(1:fileLengths(k),k) = x;  % Append new record
       into Input Matrix
end
```

```
%%
% So now we have all of the benign records in "Data".  Let's
    truncate the
% inputs to the minimum length to ensure uniform record
   lengths.

smallest = min(fileLengths);
AtkData=AtkData(1:smallest,:);

% Now check each file to verify it is the same length
  % refSize = sum(Data(:,1)>=0);  % Deprecated code to count
      number of
  % elements != -1 in matrix column

refSize = smallest;
difFileLengths = 0;
for i = 1:numFiles
    if size(AtkData(:,i)) ~= refSize
        fprintf('***WARNING:␣THIS␣WARNING␣SHOULD␣NOT␣APPEAR.
           ␣NOT␣ALL␣INPUT␣FILES␣ARE␣OF␣SAME␣SIZE␣!!␣\n')
        difFileLengths = 1;
    end
end
%% 2b. Now we have ingested our real-life data.  Let's put
   that into the right data format for the rest of the
   script:
%%
benignCases = size(BenData,2);
attackCases = size(AtkData,2);  % SIZE for array, WIDTH for
   table
totalCases = benignCases + attackCases;

%% 2c. Pick random benign prior window
```

```
%%
benignPriorSelectors = randi([1 benignCases-1],1,N);
randomWindow = zeros(2,N);
%% 2d. Pick either attack or benign window
% Generate random integer vectors for attack and benign
    cases.
%
% NOTE THAT THIS IS THE MORE DIFFICULT CASE OF DIFFERENCING
    AGAINST ANOTHER
% WINDOW RATHER THAN THE PREVIOUS (t-1)

for i = 1:N
typeWindow = randi(2);
    if typeWindow == 1 %Benign case
        randomWindow(1,i) = 0;
        randomWindow(2,i) = benignPriorSelectors(i)+1; %
            choose the next benign window
    else
        randomWindow(1,i) = 1; % attack case
        randomWindow(2,i) = randi(attackCases);
    end
end
%%
% Now we have picked a benign prior window at random, and
    then randomly
% chosen an attack or the subsequent benign case as our "
    current" window.
%
% * This window choices are stored in the 2xN randomWindow
    array
% * The benign prior case is stored in the "
    benignPriorSelectors" 1xN array
%% 2e. Populate the data series
%% Benign

% Assign the parameterization we are using for STABLE.  We
   will use the param = 0 setting;
param = 0;
% Create zero arrays
currentWindowData = zeros(L,N);
benignPriorData = currentWindowData;

% Use the randomized source tables to populate the Current
   Window and Benign Prior Window data arrays
for i = 1:N
 % populate the benign prior data based on "
    benignPriorSelectors" randomized choices
    benignPriorData(:,i) = BenData(:,benignPriorSelectors(i)
        );
 % Repeat for unknown current window. First, determine the
    data source based on atk or benign case:
  if randomWindow(1,i) == 1   % if attack case, choose sampel
      randomly from attack cases
      currentWindowData(:,i) = AtkData(:,randi(attackCases));
  else % benign case, choose appropriate params from benign
    cases
      currentWindowData(:,i) = BenData(:,benignPriorSelectors
         (i)+1);
  end
end
%%
% Now we've generated our smallestxN array of Benign Prior
   parameters and
% a smallestxN array of current Window Params
%% 2f. Difference Data
% Zero Arrays
```

```
%%
currentDiff = zeros(1+L,N);  % extra space to copy benign or
    attack case
%%
% Difference the Current Window

for i = 1:N
    currentDiff(2:L+1,i) = sort(currentWindowData(:,i))-sort
        (benignPriorData(:,i));
    currentDiff(1,i) = randomWindow(1,i); % copy Benign ==
        0, Attack == 1 designato
end
%% 2g.  Transform and save into Array
% Generate output array.  Start with the Benign Case (note:
   the attack and benign
% cases were separated for troubleshooting; not required,
   just the way it ended
% up).

dataOutput = zeros(12+2*L,2*N);
s1 = ones(L,1);  % a ones matrix for weighting functions for
    the myriad
%% ** Begin Tuning Parameters **

kappa_myriad = 10;
kappa_levy = 0.02;
kappa_levy2 = 1;
kappa_levy3 = 4;
%% ** End Tuning Params **

% Apply test statistics to data.
for i = 1:N % benign and master index
    dataOutput(1,i) = currentDiff(1,i); % Mark as attack or
```

```
        benign Case
      % First, the Sample Myriad, a function call to STABLE
         by Jonathan Nolan
    dataOutput(2,i) = myriadfilter(currentDiff(:,i),s1,
       kappa_myriad);
    dataOutput(3,i) = LevyLoc(currentDiff(:,i),kappa_levy);
       %LLE
    dataOutput(4,i) = ZOLest(currentDiff(:,i)); %ZOLs
  % Must two-step this calculation to allow removal of -Inf
     values from ln(packet count) iff packetCount == 0
    t = log(abs(currentDiff(:,i)));
    dataOutput(6,i) = exp(sum(t(t~=-Inf))*1/L); %ZOP
    dataOutput(5,i) = dataOutput(6,i)*(sum(currentDiff(:,i)-
       dataOutput(4,i))); %ZOD
    dataOutput(7,i) = mean(currentDiff(:,i)); % Mean
    dataOutput(8,i) = median(currentDiff(:,i)); % Median
    dataOutput(9,i) = var(currentDiff(:,i)); % Variance
    dataOutput(10,i) = -96;
    dataOutput(11:10+L,i) = currentWindowData(:,i);
% %     dataOutput(11:10+L,j) = attackData(:,i);
   dataOutput(10+L+1,i) = -96;
   dataOutput(10+L+2:11+L+L,i) = benignPriorData(:,i);
% 2 other LLE for different values of kappa
    dataOutput(12+L+L,i) = LevyLoc(currentDiff(:,i),
       kappa_levy2);
    dataOutput(13+L+L,i) = LevyLoc(currentDiff(:,i),
       kappa_levy3);
% Calc the non-differenced ZOP for comparison
    t = log(abs(currentWindowData(:,i)));
    dataOutput(14+2*L,i) = exp(sum(t(t~=-Inf))*1/L);
   % Calc the non-differenced ZOD for comparison
   dataOutput(15+2*L,i) = dataOutput(14+2*L,i)*(sum(
      currentWindowData(:,i)-dataOutput(4,i)));
    % And the abs(ZOD) for grins
    dataOutput(16+2*L,i) = dataOutput(6,i)*(sum(abs(
        currentDiff(:,i)-dataOutput(4,i))));
        % And the alternate differenced ZOP using the PROD
            form vice SUM
    dataOutput(17+2*L,i) = (prod(abs(currentDiff(:,i))))^(1/L
        );
        % The non-differenced ZOL for comparison
    dataOutput(18+2*L,i) = ZOLest(currentWindowData(:,i));
if i==5000
     disp('5000␣Trials␣Done');
elseif i == 10000
     disp('10,000␣Trials␣done')
elseif i == 15000
     disp('15,000␣Trials␣Done')
end
end

%% 2e.  Generate ROCs
%%  ### NOTE that many ROCs were removed to reduce the
    amount of published code
[ndZOLx,ndZOLy,T,AUC_ndZOL] = perfcurve(dataOutput(1,:),
    dataOutput(18+2*L,:),1);

%% 2f. Calc Hoeffding Inequality for Threshold
%%
CFAR = 0.1  % Set the planned False Alarm Rate
thresholds_realdata = zeros(1,N);
for i = 1:N
     thresholds_realdata(i) = (-log(CFAR)*(0.5*(max(
         benignPriorData(:,i))-min(benignPriorData(:,i)))^2))
         ^(0.5);
end
```

```
%% 3. THIS IS THE GENERATED DATA CASE
% * Import the fitted stable parameters for the attack and
    benign analyzed datasets.
% (Code from MATLAB).
%%
% NOTE THAT THESE Reads "importfileBen and importFileAtk"
    were generated for other input files...so they may not
    apply to all data file reads.  The point is that we are
    importing 4-parameter STABLE ML fits for all ATTACK or
    BENIGN cases from Excel files to create our attack and
    benign pools of data.  These fits will be used to
    generate data samples, which are then processed in a
    manner similar to above.

benignDistros = importfileBen('D:\MAWILab\20160428\3secSegs\
    BenignSegs\3s_0.5overlap_4msSubWindow\{0}_t_01000_497
    .5601941747573_500.5601941747573.txt.xlsx','Sheet1',2,5);
% Have to use this Attack file because it has been edited to
     remove transition and non-attack samples
attackDistros = importfileAtk('D:\MAWILab\20160428\3secSegs\
    AttackSegs\20160428_Attack_3s_4ms\{0}_t_0100_565
    .2415143603133_568.2415143603133.txt.xlsx','Sheet1',2,5);
benignCases = width(benignDistros);
attackCases = width(attackDistros);
totalCases = benignCases + attackCases;
inputCases = zeros(5,totalCases);
%%

inputCases(1:4,1:benignCases) = table2array(benignDistros);
inputCases(1:4,benignCases+1:totalCases) = table2array(
    attackDistros);
```

```
inputCases(5,benignCases+1:totalCases) = 1;
%% 3a. Pick random benign prior window

benignPriorSelectors = randi([1 benignCases-1],1,N);
randomWindow = zeros(2,N);
%% 3b. Pick either attack or benign window
% Generate random integer vectors for attack and benign
    cases.
%
% NOTE THAT THIS IS THE MORE DIFFICULT CASE OF DIFFERENCING
    AGAINST ANOTHER
% WINDOW RATHER THAN THE PREVIOUS (t-1)

for i = 1:N
typeWindow = randi(2);
    if typeWindow == 1 %Benign case
        randomWindow(1,i) = 0;
        randomWindow(2,i) = benignPriorSelectors(i)+1; %
            choose the next benign window
    else
        randomWindow(1,i) = 1; % attack case
        randomWindow(2,i) = randi(attackCases);
    end
end
%%
% Now we have picked a benign prior window at random, and
    then randomly
% chosen an attack or the subsequent benign case as our "
    current" window.
%
% * This window choices are stored in the 2xN randomWindow
    array
% * The benign prior case is stored in the "
```

```
    benignPriorSelectors" 1xN array
%% 3c. Generate stable data series
%% Benign

% Assign the parameterization we are using for STABLE.  We
    will use the param = 0 setting;
param = 0;
% Create zero arrays
currentWindowData = zeros(L,N);
benignPriorData = currentWindowData;

% Now drop stable Estimates in Each
tic
for i = 1:N
 % populate the benign prior data
     theta = table2array(benignDistros(1:4,
         benignPriorSelectors(i)));
   % Generate L samples from MLE stable fit using STABLE
       vice MATLAB
     benignPriorData(:,i) = round(stablernd(L,theta,param));
 % Repeat for unknown current window. First, determine the
     theta based on atk or benign case:
 if randomWindow(1,i) == 1   % if attack case, choose from
     attack cases
       theta = (table2array(attackDistros(1:4,randomWindow(2,i
          ))));
 else % benign case, choose appropriate params from benign
     cases
       theta = (table2array(benignDistros(1:4,randomWindow(2,i
          ))));
   % Generate L samples from MLE stable fit using STABLE
       vice MATLAB
 end
```

```
    % Then populate the currentWindowData based on appropriate
        theta
    currentWindowData(:,i) = round(stablernd(L,theta,param));
end
timetoGenerate_N_Stable_RVs = [toc,N]
%%
% Now we've generated a 4xN array of Benign Prior parameters
    and a 4xN array
% of current Window Params
%% 3d. Difference Data
% Zero Arrays
%%
currentDiff = zeros(1+L,N); % extra space to copy benign or
    attack case
%%
% Difference the Current Window

for i = 1:N
    currentDiff(2:L+1,i) = sort(currentWindowData(:,i))-sort
        (benignPriorData(:,i));
    currentDiff(1,i) = randomWindow(1,i); % copy Benign ==
        0, Attack == 1 designato
end
%% 3e.  Transform and save into Array
% Generate output array.  Start with the Benign Case (note:
   the attack and benign
% cases were separated for troubleshooting; not required,
   just the way it ended
% up).
% ### The remaining code for the Generated Data case is
   duplicative of the code for the Real Data.  Truncating
   for space constraints ###
```

```
%% #### Called Functions for Location Estimation ####
%  #### Note that myriadFilter is a function of STABLE, by
    Jonathan Nolan

% Take a vector input and return the Levy Location
function LevyLoc = LevyLoc(sample,dispersion)
     % Inputs are a vector SAMPLE and a constant, DISPERSION

%size = length(sample);
%syms X G D
%f = symfun(((X-G)^(1.5))*exp(D^2/(2*(X-G))), [X G D]);

fun = @(G)sum(log(((abs(sample-G).^(1.5)).*exp((dispersion
    ^2)./(2*abs(sample-G))))));

G = fminbnd(fun,-max(abs(sample)),max(abs(sample)));
LevyLoc = G;

%% ZERO ORDER LOCATION Algorithm
% Import the data

function zol = ZOLest(sample)

% "sample" is vector input; "zol" is one of the multiply-
    occurring values
% in the vector and a location estimate of the vector, a
    robust equivalent
% of the sample mean
% This function implements the Zero Order Location estimate
    of Gonzalez and Arce (1997)
% For each multiply-occurring value (MOV) in the sample
    vector, find the product of the residuals for each non-
    equal sample
% This is Equation (8) in Gonzalez and Arce (1997) "Zero-
    Order Statistics: A signal processing framework for very
    impulsive processes"
% a.k.a. Zero Order Location estimate.  We did not use
    Equation (12)
% [product of residuals] because it hit infty for large
    sample sizes

%% Identify elements occuring more than once
% Thanks to Stephen Cobeldick for the below logic to
    identify modes in the data sample
U = unique(sample);
modes = U(1<histc(sample,unique(sample)));
results = zeros(2,length(modes));

%% Now for each mode, calculate the residual total
% For each multiply-occurring value (MOV) in the sample
    vector, find the sum of the residuals for each non-equal
    sample
 % This is Equation (8) in Gonzalez and Arce (1997) "Zero-
     Order Statistics: A signal processing framework for very
      impulsive processes"
  % a.k.a. Zero Order Location estimate
for j = 1:length(modes)
    resid = 0;  % "Zero" the residuals counter.
    % Now compare each sample value with MOV.  If thy are
       not equal, apply the difference to the sum vector
    for i = 1:length(sample)
        if modes(j) ~= sample(i)
            resid = resid + log(abs(sample(i)-modes(j)));
        end
    end
    % Update the results table with the mode value and the
        product
        results(1,j) = modes(j);
        results(2,j) = resid;
end

%% Now return the ZOL estimate location = minimum residual
    of products
[Min,Index] = min(results(2,:));
zol = results(1,Index);
end
```

Bibliography

[1] Arbor Networks, “Worldwide infrastructure security report,” Westford, MA, 2017. Available: https://www.arbornetworks.com/report/

[2] V. Paxson and S. Floyd, “Wide area traffic: the failure of poisson modeling,” *IEEE/ACM Trans. Netw.*, vol. 3, no. 3, pp. 226–244, 1995.

[3] W. Willinger, V. Paxson, and M. S. Taqqu, “Self-similarity and heavy tails: Structural modeling of network traffic,” *A practical guide to heavy tails: statistical techniques and applications*, vol. 23, pp. 27–53, 1998.

[4] A. Karasaridis and D. Hatzinakos, “Network heavy traffic modeling using/spl alpha/-stable self-similar processes,” *IEEE Trans. Commun.*, vol. 49, no. 7, pp. 1203–1214, 2001.

[5] A. Scherrer, N. Larrieu, P. Owezarski, P. Borgnat, and P. Abry, “Non-gaussian and long memory statistical characterizations for internet traffic with anomalies,” *IEEE Trans. Depend. Sec. Comput.*, vol. 4, no. 1, pp. 56–70, 2007.

[6] M. Thottan and C. Ji, “Adaptive thresholding for proactive network problem detection,” in *Proc. of the IEEE Third Int. Workshop on Syst. Manage.*, 1998, pp. 108–116.

[7] P. A. R. Kumar and S. Selvakumar, “Detection of distributed denial of service attacks using an ensemble of adaptive and hybrid neuro-fuzzy systems,” *Comput. Commun.*, vol. 36, no. 3, pp. 303–319, 2013.

[8] N. Moustafa and J. Slay, “The evaluation of network anomaly detection systems: Statistical analysis of the unsw-nb15 data set and the comparison with the kdd99 data set,” *Inform. Security J.: A Global Perspective*, vol. 25, no. 1-3, pp. 18–31, 2016.

[9] F. Simmross-Wattenberg, A. Tristan-Vega, P. Casaseca-de-la Higuera, J. I. Asensio-Perez, M. Martın-Fernandez, Y. A. Dimitriadis, and C. Alberola-Lopez, “Modelling network traffic as α–stable stochastic processes. an approach towards anomaly detection,” *Proc. VII Jornadas de Ingenierıa Telematica*, pp. 25–32, 2008.

[10] F. Simmross-Wattenberg, J. I. Asensio-Perez, P. Casaseca-de-la Higuera, M. Martin-Fernandez, I. A. Dimitriadis, and C. Alberola-Lopez, “Anomaly detection in network traffic based on statistical inference and alpha-stable modeling,” *IEEE Trans. Depend. Sec. Comput.*, vol. 8, no. 4, pp. 494–509, 2011.

[11] R. D. Pierce, “Application of the positive alpha-stable distribution,” in *Proc. of the IEEE Signal Process. Workshop on Higher-Order Statistics*, 1997, pp. 420–424.

[12] S. Painter, “Random fractal models of heterogeneity: The levy-stable approach,” *Mathematical Geology*, vol. 27, no. 7, pp. 813–830, 1995.

[13] J. P. Nolan, “Maximum likelihood estimation and diagnostics for stable distributions,” *Lévy processes: Theory and applications*, pp. 379–400, 2001.

[14] R. Feldman and M. Taqqu, *A Practical Guide to Heavy Tails: Statistical Techniques and Applications*. Boston, MA: Birkhäuser, 1998.

[15] V. A. Aalo, K. P. Peppas, and G. Efthymoglou, “Performance of CA-CFAR detectors in nonhomogeneous positive alpha-stable clutter,” *IEEE Trans. Aerosp. Electron. Syst.*, vol. 51, no. 3, pp. 2027–2038, 2015.

[16] P. Tsakalides and C. L. Nikias, “Deviation from normality in statistical signal processing: Parameter estimation with alpha-stable distributions,” *A Practical Guide to Heavy Tails: Statistical Techniques and Applications*, pp. 379–404, 1998.

[17] G. Tsihrintzis and C. Nikias, “Evaluation of fractional, lower-order statistics-based detection algorithms on real radar sea-clutter data,” *IEE Proceedings-Radar, Sonar and Navigation*, vol. 144, no. 1, pp. 29–38, 1997.

[18] H. El Ghannudi, L. Clavier, N. Azzaoui, F. Septier, and P.-A. Rolland, “α-stable interference modeling and Cauchy receiver for an IR-UWB ad-hoc network,” *IEEE Trans. Commun.*, vol. 58, no. 6, pp. 1748–1757, 2010.

[19] G. Xiaohu, Z. Guangxi, and Z. Yaoting, “On the testing for alpha-stable distributions of network traffic,” *Comput. Commun.*, vol. 27, no. 5, pp. 447–457, 2004.

[20] C. Bollmann, M. Tummala, J. McEachen, J. Scrofani, and M. Kragh, “Techniques to improve stable distribution modeling of network traffic,” in *Proc. of the 51st Hawaii Int. Conf. on Syst. Sci.*, 2018, pp. 5524–5531.

[21] S. Banerjee and M. Agrawal, “Underwater acoustic noise with generalized gaussian statistics: Effects on error performance,” in *Proc. of MTS/IEEE OCEANS13*, 2013, pp. 1–8.

[22] M. Coates and E. Kuruoğlu, “Time–frequency-based detection in impulsive noise environments using α-stable noise models,” *Signal Process.*, vol. 82, no. 12, pp. 1917–1925, 2002.

[23] J. G. Gonzalez, “Robust techniques for wireless communications in non-Gaussian environments,” Ph.D. dissertation, Dept. of Elec. Eng., Univ. of Delaware, Newark, DE, 1997.

[24] J. Park, G. Shevlyakov, and K. Kim, “Maximin distributed detection in the presence of impulsive alpha-stable noise,” *IEEE Trans. Wireless Commun.*, vol. 10, no. 6, pp. 1687–1691, 2011.

[25] A. G. Tartakovsky and A. S. Polunchenko, “Quickest changepoint detection in distributed multisensor systems under unknown parameters,” in *Proc. of 11th Int. Conf. on Inform. Fusion*, 2008, pp. 1–8.

[26] A. G. Tartakovsky, B. L. Rozovskii, R. B. Blazek, and H. Kim, “A novel approach to detection of intrusions in computer networks via adaptive sequential and batch-sequential change-point detection methods,” *IEEE Trans. Signal Process.*, vol. 54, no. 9, pp. 3372–3382, 2006.

[27] A. G. Tartakovsky and G. V. Moustakides, “State-of-the-art in Bayesian changepoint detection,” *Sequential Anal.*, vol. 29, no. 2, pp. 125–145, 2010.

[28] A. G. Tartakovsky, A. S. Polunchenko, and G. Sokolov, “Efficient computer network anomaly detection by changepoint detection methods,” *IEEE J. Sel. Topics Signal Process.*, vol. 7, no. 1, pp. 4–11, 2013.

[29] E. E. Kuruoglu, C. Molina, and W. J. Fitzgerald, “Approximation of α-stable probability densities using finite gaussian mixtures,” in *Proc. of the 9th Eur. Signal Process. Conf.*, 1998, pp. 1–4.

[30] E. E. Kuruoglu, “Analytical representation for positive α-stable densities,” in *Proc. of IEEE Int. Conf. on Acoust., Speech, and Signal Process.*, 2003, vol. 6, pp. 729–732.

[31] E. E. Kuruoglu, W. J. Fitzgerald, and P. J. Rayner, “Near optimal detection of signals in impulsive noise modeled with a symmetric α-stable distribution,” *IEEE Commun. Lett.*, vol. 2, no. 10, pp. 282–284, 1998.

[32] J. G. Gonzalez and G. R. Arce, “Weighted myriad filters: A robust filtering framework derived from alpha-stable distributions,” in *Proc. of IEEE Int. Conf. on Acoust., Speech, and Signal Process.*, 1996, vol. 5, pp. 2833–2836.

[33] J. G. Gonzalez, J. L. Paredes, and G. R. Arce, “Zero-order statistics: A mathematical framework for the processing and characterization of very impulsive signals,” *IEEE Trans. Signal Process.*, vol. 54, no. 10, pp. 3839–3851, 2006.

[34] G. R. Arce, J. G. Gonzalez, and Y. Li, “Weighted myriad filters,” in *Nonlinear signal and image processing: Theory, methods, and applications*, K. E. Barner and G. R. Arce, Eds. Washington, D.C.: CRC Press, 2003, pp. 151–194.

[35] A. Tirumala, “Iperf: The tcp/udp bandwidth measurement tool,” 1999. Available: https://sourceforge.net/projects/iperf/files/iperf/

[36] D. Dittrich. The DoS Project’s ‘trinoo’ distributed denial of service attack tool. University of Washington, October 21, 1999. [Online]. Available: https://staff.washington.edu/dittrich/misc/trinoo.analysis.txt

[37] Apache software foundation, “Apache JMeter,” 2010. Available: http://jmeter.apache.org/

[38] G. Combs *et al.*, “Wireshark,” 1998. Available: http://www.wireshark.org

[39] MATLAB, *version 9.2 (R2017a)*. Natick, Massachusetts: The MathWorks Inc., 2017. Available: https://www.mathworks.com/products/matlab.html

[40] G. Samorodnitsky and M. S. Taqqu, *Stable non-Gaussian random processes: Stochastic models with infinite variance*. New York, NY: CRC press, 1994.

[41] J. P. Nolan, *Stable Distributions - Models for Heavy Tailed Data*. Boston, MA: Birkhäuser, 2018, in progress, Chapter 1 online at http://fs2.american.edu/jpnolan/www/stable/stable.html.

[42] J. P. Nolan, “Modeling financial data with stable distributions,” *Handbook of Heavy Tailed Distributions in Finance, Handbooks in Finance: Book*, vol. 1, pp. 105–130, 2003.

[43] G. A. Tsihrintzis, “Statistical modeling and receiver design for multi-user communication networks,” *A Practical Guide to Heavy Tails: Statistical Techniques and Applications*, 1998.

[44] S. M. Kay, *Fundamentals of statistical signal processing, volume 1: Estimation theory*. Boston, MA: Prentice Hall Signal Processing Series, AV Oppenheim, Ed. Prentice Hall PTR, 1993.

[45] C. L. Nikias and M. Shao, *Signal Processing with Alpha-stable Distributions and Applications*. New York, NY: Wiley-Interscience, 1995.

[46] D. Panchenko. 18.443 Statistics for Applications. Massachusetts Institute of Technology: MIT OpenCourseWare. [Online]. Available: https://ocw.mit.edu

[47] D. J. Weller-Fahy, B. J. Borghetti, and A. A. Sodemann, “A survey of distance and similarity measures used within network intrusion anomaly detection,” *IEEE Commun. Surveys Tut.*, vol. 17, no. 1, pp. 70–91, 2015.

[48] M. M. Deza and E. Deza, “Encyclopedia of distances,” in *Encyclopedia of Distances*. Springer, 2009, pp. 1–583.

[49] S. M. Kay, *Fundamentals of statistical signal processing, volume 2: Detection theory*. Boston, MA: Prentice Hall Signal Processing Series, AV Oppenheim, Ed. Prentice Hall PTR, 1998.

[50] M. H. Bhuyan, D. K. Bhattacharyya, and J. K. Kalita, "Network anomaly detection: methods, systems and tools," *IEEE Commun. Surveys Tut.*, vol. 16, no. 1, pp. 303–336, 2014.

[51] N. Hoque, M. H. Bhuyan, R. C. Baishya, D. Bhattacharyya, and J. K. Kalita, "Network attacks: Taxonomy, tools and systems," *J. of Network and Comput. Applications*, vol. 40, pp. 307–324, 2014.

[52] S. M. Kay, *Fundamentals of statistical signal processing, volume 3: Practical algorithm development*. Boston, MA: Pearson Education, 2013.

[53] J. Ilow and D. Hatzinakos, "Analytic alpha-stable noise modeling in a poisson field of interferers or scatterers," *IEEE Trans. Signal Process.*, vol. 46, no. 6, pp. 1601–1611, 1998.

[54] R. Fontugne, P. Borgnat, P. Abry, and K. Fukuda, "MAWILab: Combining diverse anomaly detectors for automated anomaly labeling and performance benchmarking," in *Proc. of CoNEXT 2010*, 2010, p. 8.

[55] Canadian Institute for Cybersecurity. Intrusion Detection Evaluation Dataset. Information Security Center of Excellence. [Online]. Available: http://www.unb.ca/cic/research/datasets/ids.html

[56] A. Shiravi, H. Shiravi, M. Tavallaee, and A. A. Ghorbani, "Toward developing a systematic approach to generate benchmark datasets for intrusion detection," *Comput. & Security*, vol. 31, no. 3, pp. 357–374, 2012.

[57] 2010 Mid-Atlantic Collegiate Cyber Defense Competition. Hosted by Netresec. [Online]. Available: https://www.netresec.com/?page=MACCDC

[58] University of Texas Center for Information Assurance and Security. "Team Preparation Guide". [Online]. Available: http://nccdc.org/files/CCDCteamprepguide.pdf

[59] K. S. Shanmugan and A. M. Breipohl, *Random signals: Detection, estimation, and data analysis*. New York, NY: John Wiley & Sons, 1988.

[60] C. C. Aggarwal, *Outlier analysis*. New York, NY: Springer, 2013.

[61] P. Chhabra, C. Scott, E. D. Kolaczyk, and M. Crovella, "Distributed spatial anomaly detection," in *Proc. of the IEEE 27th Conf. on Comput. Commun.*, 2008.

[62] R. J. Hyndman and G. Athanasopoulos. (2014). *Forecasting: Principles and practice*. OTexts. [Online]. Available: https://www.otexts.org/fpp

[63] C. A. Bollmann, M. Tummala, and J. McEachen, "Representation of positive alpha-stable network traffic through levy mixtures," unpublished.

[64] K. Górska and K. Penson, "Lévy stable two-sided distributions: Exact and explicit densities for asymmetric case," *Physical Rev. E*, vol. 83, no. 6, p. 061125, 2011.

[65] T. K. Pogány and S. Nadarajah, "Remarks on the stable $S_\alpha(\beta, \gamma, \mu)$ distribution," *Meth. and Comp. in App. Prob.*, vol. 17, no. 2, pp. 515–524, 2015.

[66] C. D. Hardin, "Skewed stable variables and processes," Center for Stochastic Processes, Chapel Hill, NC, Tech. Rep. #79, Sep. 1984. Available: http://www.dtic.mil/dtic/tr/fulltext/u2/a150549.pdf

[67] J. G. Gonzalez and G. R. Arce, "Statistically-efficient filtering in impulsive environments: Weighted myriad filters," *EURASIP J. on Advances in Signal Process.*, vol. 2002, no. 1, p. 363195, 2002.

[68] E. E. Kuruoglu, "Density parameter estimation of skewed α-stable distributions," *IEEE Trans. Signal Process.*, vol. 49, no. 10, pp. 2192–2201, 2001.

[69] G. R. Arce. (2005). *Nonlinear signal processing: A statistical approach*. Wiley-Interscience. [Online]. Available: http://onlinelibrary.wiley.com/book/10.1002/0471691852

[70] M. H. Bhuyan, D. K. Bhattacharyya, and J. K. Kalita, *Network Traffic Anomaly Detection and Prevention: Concepts, Techniques, and Tools*. Springer, 2017.

[71] P. Tsakalides, F. Trinic, and C. L. Nikias, "Performance assessment of cfar processors in pearson-distributed clutter," *IEEE Trans. Aerosp. Electron. Syst.*, vol. 36, no. 4, pp. 1377–1386, 2000.

[72] J. Whitney and P. Delforge, "Data center efficiency assessment," The National Resources Defense Council, New York, NY, Tech. Rep. IP-14-08-A, Aug. 2014. Available: https://www.nrdc.org/sites/default/files/data-center-efficiency-assessment-IP.pdf

[73] C. Callegari and M. Pagano, "A novel bivariate entropy-based network anomaly detection system," in *Int. Conf. on Security, Privacy and Anonymity in Computation, Commun. and Storage*. Springer, 2017, pp. 168–179.

[74] X. Ma and Y. Chen, "Ddos detection method based on chaos analysis of network traffic entropy," *IEEE Commun. Lett.*, vol. 18, no. 1, pp. 114–117, 2014.

[75] Z. Wu, L. Zhang, and M. Yue, “Low-rate dos attacks detection based on network multifractal,” *IEEE Trans. Depend. Sec. Comput.*, vol. 13, no. 5, pp. 559–567, 2016.

[76] C. Wang, T. N. Miu, X. Luo, and J. Wang, “Skyshield: A sketch-based defense system against application layer ddos attacks,” *IEEE Trans. Inf. Forensics Security*, 2017.

[77] G. Yu, C. Li, and J. Zhang, “A new statistical modeling and detection method for rolling element bearing faults based on alpha–stable distribution,” *Mech. Syst. and Signal Process.*, vol. 41, no. 1-2, pp. 155–175, 2013.

[78] S. T. Zargar, J. Joshi, and D. Tipper, “A survey of defense mechanisms against distributed denial of service (ddos) flooding attacks,” *IEEE Comm. Surveys Tut.*, vol. 15, no. 4, pp. 2046–2069, 2013.

[79] R. Ben Basat, G. Einziger, R. Friedman, M. C. Luizelli, and E. Waisbard, “Constant time updates in hierarchical heavy hitters,” in *Proc. of the Conf. of the ACM Special Interest Group on Data Commun.*, 2017, pp. 127–140.

[80] S. Bhatia, “Ensemble-based model for ddos attack detection and flash event separation,” in *Proc. of the IEEE Future Technol. Conf.* IEEE, 2016, pp. 958–967.

[81] S. Shanbhag and T. Wolf, “Accurate anomaly detection through parallelism,” *IEEE Netw.*, vol. 23, no. 1, pp. 22–28, 2009.

[82] J. Kittler, M. Hatef, R. P. Duin, and J. Matas, “On combining classifiers,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 20, no. 3, pp. 226–239, 1998.

THIS PAGE INTENTIONALLY LEFT BLANK

Initial Distribution List

1. Defense Technical Information Center
   Ft. Belvoir, Virginia

2. Dudley Knox Library
   Naval Postgraduate School
   Monterey, California